PUBLIC
SQL Anywhere Server
Document Version: 17.01.0 – 2021-10-15

SQL Anywhere Database Administration

THE BEST RUN SAP

Content

1 SQL Anywhere Server - Database Administration

This book describes how to run, manage, and configure SQL Anywhere databases.

It describes database connections, the database server, database files, backup procedures, security, high availability, as well as administration utilities and options.

In this section:

1.1 Database Connections

Several tools and methods are provided for connecting to a database.

In this section:

1.1.1 Connection Basics

Learn how to quickly connect to and disconnect from a database.

In this section:

1.1.1.1 Connecting to a Database (Interactive SQL, SQL Central)

Connect to a local or remote database using the *Connect* window.

Context

This task describes how to connect to a database using the *Connect* window. The *Connect* window provides options to connect to a running database, as well as to start a database and then connect to it. The examples for connecting to a running database assume that a database is running.

Some choices in this task teach you how to connect to a database that is already started. For these choices, it is assumed that the database is already started. If you attempt the steps on a database that is not started, the steps will fail.

Procedure

1. Open the *Connect* window.
2. In the *Authentication* dropdown list, click *Database*.
3. In the *User ID* and *Password* fields, type a user name and password.
4. Choose one of the following options:

Option	Action
Start and connect to a database on your local computer	This option starts a database on a personal database server on your local computer. 1. In the *Action* dropdown list, click *Start and connect to a database on this computer*. 2. In the *Database file* field, specify the database file path, file name, and file extension. For example, type C:\Users\Public\Documents\SQL Anywhere 17\Samples\demo.db. 3. To create a database name that is different from the file name for subsequent connections, type a name in the *Database name* field. Do not specify a file path or extension. 4. In the *Server name* field, specify a name for the database server. For example, type ***demo17***. Since a computer system can run multiple database servers, always specify the database server name (ServerName=`server-name`) when connecting to a database.
Connect to a database that is running on your local computer	This option connects to a database that is running on a personal database server on your local computer. 1. In the *Action* dropdown list, click *Connect to a running database on this computer* to connect to a database that is running on your computer. 2. In the *Server name* field, type the name of the running database server. For example, type ***demo17***. 3. Since a computer system can run multiple database servers, always specify the database server name (ServerName=`server-name`) when connecting to a database. 4. In the *Database name* field, type the name of the database. For example, type ***demo***.
Connect to a database that resides on another computer	This option connects to a database that is running on a network database server. 1. In the *Action* dropdown list, click *Connect to a running database on another computer* to connect to a database that is running on another computer. 2. In the *Host* field, type the host name of the computer where the database server is running. 3. In the *Server name* field, type the name of the server. For example, type ***demo17***. Since a computer system can run multiple database servers, always specify the database server name (ServerName=server-name) when connecting to a database. 4. In the *Database Name* field, type the name of the database. For example, type ***demo***.
Connect with an ODBC Data Source	This option connects to a database using an ODBC Data Source. 1. In the *Action* dropdown list, click *Connect with an ODBC Data Source*. 2. Do one of the following: • Click *ODBC Data Source name* and type the name of the data source. For example, type ***SQL Anywhere 17 Demo*** (for this data source, you must also specify the password **`sql`**). • Click *ODBC Data Source file* and type the FileDataSourceName (FILEDSN) connection parameter that references a data source held in a file. The *SQL Anywhere 17 Demo* data source is configured to start the database server and database if they are not already running. If your ODBC data source is not configured to start the database and database server, then you must start them.
Connect with a connection string	This option connects to a database. 1. In the *Action* dropdown list, click *Connect with a connection string*.

Option	Action
	This option is useful if you have an ODBC data source and you want to specify additional or different parameters when you connect. If a connection string and an ODBC data source both specify the same connection parameter, the value from the connection string is used and the value from the data source is ignored. 2. In the *Parameters* field, type connection parameters in a semicolon-delimited list of parameter=value pairs. For example: `DSN=SQL Anywhere 17 Demo;PWD=sql;ServerName=SampleServer` The database server and the database starts and the administration tool connects to the database. The database server is named SampleServer. Since a computer system can run multiple database servers, always specify the database server name (ServerName=`server-name`) when connecting to a database. The *SQL Anywhere 17 Demo* data source is configured to start the database server and database if they are not already running. If your ODBC data source is not configured to start the database and database server, then you must start them.
Start and connect to a database on another computer	This option starts a database on a network database server and then connects to the database. 1. Start a database server on the other computer and specify the -gd or -gp database server option. **⚠ Caution** The -gd or -gp database server options allow any user to start a database on the server. For example, run the following command to start a database server: `dbsrv17 -n myserver -gd all` 2. 1. In the *Action* dropdown list, click *Start and connect to a database on another computer*. 2. In the *Database File* field, type the path and extension of the database file, relative to the database server. For example, type ***C:\Users\Public\Documents\SQL Anywhere 17\Samples\demo.db***. 3. To create a database name that is different from the file name for subsequent connections, type a name in the *Database name* field. Do not specify a file path or extension. For example, type **`mydemo`**. 4. In the *Server name* field, specify a name for the server. For this example, use the server name **`myserver`** that was specified above. Since the host computer may be running several database servers, it is recommended that you always specify the database server name. 5. In the *Host* field, type the host name of the computer where the database server is running.

5. Optional: To get a copy of the connection string that is created with this window, click *Tools* > *Copy Connection String to Clipboard*.
6. Click *Connect*.

Results

The administration tool connects to the database.

Related Information

Tutorial: Connecting to the Sample Database [page 21]

1.1.1.2 Connecting to Databases on Local and Remote Servers

Connect to a database using a connection string.

Prerequisites

When you connect to the database server, you actually connect to a database.

Typically, you start a database server with one or more databases before the first client connects to it. For example, the following command starts the SQL Anywhere 17 sample database on a database server (replace `server-name` with a suitable server name):

```
dbsrv17 -n server-name "C:\Users\Public\Documents\SQL Anywhere
         17\Samples\demo.db"
```

If the database server has not already been started, then it must be started. If the server computer and the client computer are the same, then the server can be started automatically as part of the client connection process.

Similarly, if the database has not already been started on the database server, then it must be started. If the server computer and the client computer are the same then a database can be started automatically as part of the client connection process. Under certain conditions, a database can also be started on the remote server computer by the client.

> → Tip
>
> Since a computer system can run multiple database servers, always specify the database server name (ServerName=`server-name`) when connecting to a database.

Context

Each of the following choices describes a unique connection scenario. Choose the one that best suits your situation. Interactive SQL is used for demonstration.

Procedure

- Connect to a database running on your local computer.

  ```
  dbisql -c "UID=user-id;PWD=password;ServerName=server-name;DBN=database-name"
  ```

  Both server and database must already be started.
- Start and connect to a database on your local computer.

  ```
  dbisql -c "UID=user-id;PWD=password;ServerName=server-name;DBF=database-file"
  ```

  If the specified database server is not already running, it is started. If the database is not already started, then it is started on the server.
- Connect to a database running on a network database server.

  ```
  dbisql -c "UID=user-ID;PWD=password;ServerName=server-name;Host=host-name"
  ```

  If the specified server is not already running, it is will not be started. If the server is running, you will connect to the first database that was started on the server. Use DatabaseName (DBN) to specify which database to connect to.
- Connect using an ODBC data source.

  ```
  dbisql -c "DSN=SQL Anywhere 17 Demo;PWD=sql"
  ```

  If the specified database server is not already running, it is started. If the database is not already started, then it is started on the server.

  The SQL Anywhere 17 Demo data source contains the server name (ServerName) and the database file name (DBF).

  You do not need to specify a user ID for this connection because the data source already contains this information (UID=DBA). You must specify the password since it is not included in the data source.
- Connect using an ODBC data source with additional connection parameters.

  ```
  dbisql -c "DSN=SQL Anywhere 17 Demo;PWD=sql;ServerName=SampleServer"
  ```

  If the specified database server is not already running, it is started. If the database is not already started, then it is started on the server.

  The SQL Anywhere 17 Demo data source contains the server name (ServerName) and the database file name (DBF). The server name is overridden by the ServerName connection parameter that is provided on the command line.

  You do not need to specify a user ID for this connection because the data source already contains this information (UID=DBA). You must specify the password since it is not included in the data source.

- Start and connect to a database on another computer.
  a. Start a database server on the other computer and specify the -gd database server option. Specify the -gp database server option to set the default page size larger than 4096. Database files with a page size larger than the specified page size cannot be started.

     ```
     dbsrv17 -n SampleServer -gd all -gp 8k
     ```

     > ⚠ Caution
     >
     > The `-gd all` database server option allows any user to start a database on the server.

  b. Run the following command (replace `host-name` with the host name of the computer running the database server):

     ```
     dbisql -c "Host=host-name;Server=SampleServer;DBF=C:\Users\Public\Documents
     \SQL Anywhere
              17\Samples\demo.db;UID=DBA;PWD=sql"
     ```

     The database file resides on the server computer.

Results

Interactive SQL connects to the database.

In this section:

1.1.1.2.1 Connections on UNIX/Linux

The Interactive SQL utility (dbisql) and the sample database are used to demonstrate how to start and connect to a database.

In this section:

Related Information

UNIX and Linux Environment Variables [page 567]

The SQL Anywhere Sample Database (demo.db)
Security: Passwords [page 1685]
Sourcing sa_config.sh, sample_config32.sh, and sample_config64.sh [UNIX/Linux] [page 568]

1.1.1.2.1.1 Connecting to the Sample Database on UNIX/Linux

Connect to the SQL Anywhere sample database and explore SQL Anywhere on UNIX and Linux.

Prerequisites

The following procedure uses Interactive SQL to connect to the database. Ensure that the Java Runtime Environment (JRE) is installed.

In a typical UNIX or Linux installation, the SQL Anywhere software, including the sample database, is installed into a directory that you do not have permissions to write to. Before you begin, in a terminal window, change to a writable folder. Run the following command to copy the sample database file into the folder:

```
cp "$SQLANYSAMP17/demo.db" .
```

In a terminal window, run the following command to start the sample database on a database server:

```
dbeng17 demo.db
```

Procedure

1. In a terminal window, start Interactive SQL:

   ```
   dbisql
   ```

   The *Connect* window appears by default.
2. Provide the following information in the *Connect* window:
   a. In the *User ID* field, type **DBA**.
   b. In the *Password* field, type **sql**.
   c. In the *Action* dropdown list, click *Connect to a running database on this computer*.
   d. In the *Server name* field, type ***demo***.
   e. Optional: To get a copy of the connection string that is created with this window, click *Tools* > *Copy Connection String to Clipboard*.
   f. Click *Connect*.

Results

Interactive SQL starts and connects to the sample database.

Related Information

The SQL Anywhere Sample Database (demo.db)
ODBC Data Sources on UNIX/Linux [page 126]
Connecting to the Sample Database on macOS [page 14]

1.1.1.2.1.2 Connecting to the Sample Database on macOS

Connect to the SQL Anywhere sample database and explore SQL Anywhere on macOS.

Prerequisites

- In the *Finder*, locate the SQL Anywhere sample database and copy the file to a location where you have read and write access. By default, the sample database is located in `/Applications/SQLAnywhere17/demo.db`.
- Start a database server. In the *Finder*, double-click *DBLauncher*. By default, *DBLauncher* is located in `/Applications/SQLAnywhere17`.
  1. In the *Database* field, browse to the location of the SQL Anywhere sample database. (For example, */Applications/SQLAnywhere17/System/demo.db*.)
  2. In the *Server name* field, type ***demo***.
  3. Click *Local Server*.
     The *Local Server* option does not allow client/server communications over a network.
  4. Click *Start* to start a personal database server named ***demo***.

Procedure

1. In the *Finder*, double-click *Interactive SQL* in `/Applications/SQLAnywhere17`.
2. Connect to the SQL Anywhere sample database by providing the following information in the *Connect* window:
   - In the *Authentication* dropdown list, click *Database*.
   - In the *User ID* field, type **`DBA`**.
   - In the *Password* field, type **`sql`**.

- In the *Action* dropdown list, click *Connect to a running database on this computer* to connect to the database that is running on your computer.
- In the *Server name* field, type ***demo***.
- In the *Database name* field, type ***demo***.
- Optional: To get a copy of the connection string that is created with this window, click *Tools* > *Copy Connection String to Clipboard*.
- Click *Connect*.

Results

Interactive SQL starts and connects to the sample database.

Related Information

The SQL Anywhere Sample Database (demo.db)
Connecting to the Sample Database on UNIX/Linux [page 13]
Creating an ODBC Data Source on macOS [page 121]

1.1.1.3 Managing Connected Users (SQL Central)

Manage all users connected to the database by using SQL Central.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

s

You must have the DROP CONNECTION, MONITOR, or SERVER OPERATOR system privilege.

Context

You can also view properties of connected users using SQL Central, and you can disconnect them.

You can also view a list of all users connected to a database from the *Connections* widget in the SQL Anywhere Monitor.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. Display a list of all users connected to the database.
   a. Select the database in the left pane, and click the *Connections* tab in the right pane.

      This tab displays all users currently connected to the database, regardless of the application that they used to connect (SQL Central, Interactive SQL, or a custom client application).
3. Inspect the properties of a user's connection to a database.
   a. On the *Connections* tab, right-click the user and then click *Properties*.
   b. Review the properties for the user and click *OK*.

Results

The properties of the connected users are reviewed.

Next Steps

You can disconnect a user from the database.

In this section:

Disconnecting from a Database (SQL Central) [page 17]
Disconnect a user from a database by using SQL Central.

Disconnecting from a Database (SQL) [page 18]
Disconnect from a database by using the DISCONNECT statement or other users by using the DROP CONNECTION statement.

Related Information

Managing Widgets [page 1280]

1.1.1.3.1 Disconnecting from a Database (SQL Central)

Disconnect a user from a database by using SQL Central.

Prerequisites

You do not need any privileges to disconnect you own connection from the database.

You must have the DROP CONNECTION system privilege to disconnect other users.

Procedure

Connect to the database.

Option	Action
Disconnect the current user	Click *File* > *Disconnect*.
Disconnect another user	Click the *Connections* tab in the right pane. Right-click the user and then click *Disconnect*.

Results

The user is disconnected from the database.

Related Information

Disconnecting from a Database (SQL) [page 18]
Closing a Connection Using the Monitor [page 1283]

1.1.1.3.2 Disconnecting from a Database (SQL)

Disconnect from a database by using the DISCONNECT statement or other users by using the DROP CONNECTION statement.

Prerequisites

You do not need any privileges to disconnect the **current user** from the database.

You must have the DROP CONNECTION system privilege to disconnect other users.

Procedure

Connect to the database.

Option	Action
Disconnect the current user	Execute a DISCONNECT statement.
Disconnect another user	Use the sa_conn_info system procedure to determine the connection ID of the user you want to disconnect. Execute a DROP CONNECTION statement.

Results

The user is disconnected from the database.

Example

The following statement disconnects the current connection, conn1, in Interactive SQL:

```
DISCONNECT conn1;
```

The following statement shows how to use DISCONNECT in Embedded SQL:

```
EXEC SQL DISCONNECT :conn-name
```

The following statement drops connection number 4:

```
DROP CONNECTION 4;
```

Related Information

Disconnecting from a Database (SQL Central) [page 17]
Closing a Connection Using the Monitor [page 1283]
DISCONNECT Statement [ESQL] [Interactive SQL]
DROP CONNECTION Statement
sa_conn_info System Procedure

1.1.1.4 Connection Profiles (SQL Central)

To save time and simplify the connection process when using SQL Central, you can create a connection profile to save the connection parameters for each database.

When you first connect to a database server or database, you specify a user ID, password, and other connection parameters. This information must be entered again when you make subsequent connections. Create a connection profile to save the connection parameters for a database in SQL Central. Connection profiles are specific to SQL Central. If you are building an ODBC application, you can use ODBC data sources to achieve functionality similar to connection profiles.

To use and manage connection profiles, in SQL Central click *Connections* > *Connection Profiles* to open the *Connection Profiles* window. Use the *Connection Profiles* window to:

- Create a new connection profile.
- Set a profile to connect automatically when SQL Central is started.
  If you typically use the same connection profile with SQL Central, you can configure SQL Central so that by default it uses your connection profile. When you start SQL Central, it uses the connection profile to connect to your database automatically.
- Edit an existing connection profile.
- Connect using a connection profile.
- Delete a connection profile.
- Import and export a connection profile.
  Share connection profiles with users on other computers or back up your connection profiles, by exporting them to a file, or use a connection profile that was created elsewhere to import the connection profile.

In this section:

Creating a Connection Profile (SQL Central) [page 20]
Create a connection profile to save the connection parameters for a database in SQL Central.

Connecting with a Connection Profile (SQL Central) [page 21]
Use a connection profile to connect to a database server or database from SQL Central.

Related Information

ODBC Data Sources [page 116]

1.1.1.4.1 Creating a Connection Profile (SQL Central)

Create a connection profile to save the connection parameters for a database in SQL Central.

Context

Storing user IDs, encrypted or unencrypted passwords, and database keys in a connection profile is not recommended.

Procedure

1. In SQL Central, click *Connections* > *Connection Profiles*.
2. Click *New*.
3. In the *Name* field, type a name for the new profile.
4. Click *New connection profile* and choose the appropriate plug-in from the dropdown list.

   To base your new connection profile on an existing profile, click *Copy connection profile* and choose the profile from the *Existing connection profiles* list.
5. (Optional) To allow other users to access the profile, click *Share this connection profile with other users*. This setting is useful on multi-user platforms such as Unix.
6. Click *OK*.
7. In the *Edit Connection Profile* window, enter the required values, and then click *Save* to close the window.
8. Click *Close*.

Results

The connection profile is created and you can use it to connect to your database.

Next Steps

You can configure SQL Central to connect automatically with this connection profile when SQL Central starts.

Related Information

ODBC Data Sources [page 116]
Connecting with a Connection Profile (SQL Central) [page 21]

1.1.1.4.2 Connecting with a Connection Profile (SQL Central)

Use a connection profile to connect to a database server or database from SQL Central.

Prerequisites

A connection profile must be defined.

Procedure

1. In SQL Central, click *Connections* > *Connection Profiles*.

   The *Connection Profiles* window appears.
2. Select a profile and then click *Connect*.

   > → Tip
   >
   > To have SQL Central automatically connect using your connection profile, click *Set Startup* to change the *Use on Startup* column to *Yes*.
3. Click *Close*.

Results

Once you are connected, an icon appears in the SQL Central main window.

1.1.1.5 Tutorial: Connecting to the Sample Database

Start the sample database and connect to it to explore the data and objects.

Prerequisites

The default DBA user is named `dba` and its password is `sql`.

Context

The SQL Anywhere sample database represents a small company that makes a limited range of sports clothing. It contains internal information about the company (employees, departments, and financial data), product information (products), and sales information (sales orders, customers, and contacts). All information in the sample database is fictional.

1. Lesson 1: Connecting to the Sample Database (SQL Anywhere 17 Demo) [page 22]
   Start a database server, connect to the sample database, and display the database server messages window to view information about the database server.
2. Lesson 2: Connecting to the Database (SQL Central) [page 25]
   Connect to the sample database to try out many of the examples found in the documentation.
3. Lesson 3: Connecting to the Sample Database (SQL) [page 26]
   Start Interactive SQL and connect to the sample database by using a command prompt.
4. Lesson 4: Stopping the Database Server (dbstop utility) [page 27]
   Stop the database server by using the Stop Server utility.

Related Information

The SQL Anywhere Sample Database (demo.db)

1.1.1.5.1 Lesson 1: Connecting to the Sample Database (*SQL Anywhere 17 Demo*)

Start a database server, connect to the sample database, and display the database server messages window to view information about the database server.

Prerequisites

You must have the roles and privileges listed at the beginning of this tutorial.

Context

The sample database uses the following default user ID and password:

- User ID = DBA
- Password = sql (passwords in SQL Anywhere are case sensitive)

The sample database uses the following ODBC data source: *SQL Anywhere 17 Demo*.

- You do not need to enter a user ID because the SQL Anywhere 17 Demo data source already contains this information, but you will need to specify the password **sql**.
  In a production environment, do not store passwords in an ODBC data source.
- The database server does not need to be running before connecting since the SQL Anywhere 17 Demo data source automatically starts the database server.

Procedure

1. In Interactive SQL, open the *Connect* window.

   Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *Interactive SQL*.

   Click *SQL* > *Connect*.
2. In the *Authentication* dropdown list, click *Database*.
3. In the *User ID* field, type `dba`. In the *Password* field, type `sql`.
4. Choose one of the following options:

Action	Description
Connect using the default ODBC data source.	The *SQL Anywhere 17 Demo* ODBC data source is configured to start the database server and database if they are not already running. 1. In the *Action* dropdown list, click *Connect with an ODBC Data Source*. 2. Do one of the following: • Click *ODBC Data Source name* and type ***SQL Anywhere 17 Demo***. 3. (Optional) To get a copy of the connection string that is created with this window, click *Tools* > *Copy Connection String to Clipboard*. 4. Click *Connect*.
Start Interactive SQL and connect to the sample database:	Run the following command: `dbisql -c "UID=DBA;PWD=sql;DBF=%SQLANYSAMP17%\demo.db"` A new Interactive SQL window opens.
Start the database server and then connect to the sample database.	In the *Connect window*, complete the following fields to connect to the sample database: 1. In the *Action* dropdown list, select *Start and connect to a running database on this computer*. In the *Server Name* field, type ***demo17***. 2. In the *File name*field specify **%SQLANYWHERE*17***\demo.db 3. Click *Connect*.

On Windows, the database server appears as an icon in the system tray. You have successfully started a database server running the sample database. However, you cannot see or manipulate the data in the database yet. The database server icon is the only visible indication that anything has happened.

5. Double-click the database server icon in the system tray to display the database server messages window.

   The database server messages window displays useful information, including:

   The server name

   The name in the title bar (in this case demo17) is the **server name**. In this tutorial, you assigned the server name by using the -n server option. If you do not provide a server name, the database server is given the name of the first database started. This name can be used by applications when they connect to a database.

   The version and build numbers

   The numbers following the server name are the version and build numbers. The version number represents the specific release of SQL Anywhere, and the build number relates to the specific instance of the software that was compiled.

   Startup information

   When a database server starts, it sets aside some memory that it uses when processing database requests. This reserved memory is called the **cache**. The amount of cache memory appears in the window. The cache is organized in fixed-size **pages**, and the page size also appears in the window. In this case, the startup cache size and page size are the default values. For many purposes, including those of this tutorial, the default startup options are fine.

   Database information

   The names of the database file and its transaction log file appear in the window.

Results

A database server running the sample database is started, and the database server messages window is displayed.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Connecting to the Sample Database [page 21]

Next task: Lesson 2: Connecting to the Database (SQL Central) [page 25]

Related Information

The SQL Anywhere Sample Database (demo.db)
How to Start the Database Server [page 324]
Database Servers [page 318]
SQL Anywhere Database Server Executable (dbsrv17, dbeng17) [page 380]

1.1.1.5.2 Lesson 2: Connecting to the Database (SQL Central)

Connect to the sample database to try out many of the examples found in the documentation.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Context

SQL Central is a graphical tool that can be used as an alternative to using SQL statements and command line utilities for managing your database servers, databases, and the objects they contain.

Procedure

1. Start SQL Central. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Central*.
2. Click *Connections* > *Connect With SQL Anywhere 17*.
3. Complete the following fields to connect to the sample database:
   a. In the *User ID* field, type **DBA**.
   b. In the *Password* field, type **sql**.
   c. In the *Action* dropdown list, select *Connect to a running database on this computer*.
   d. In the *Server Name* field, type ***demo17***.

      Since a computer system can run multiple database servers, always specify the database server name (ServerName=`server-name`) when connecting to a database.
4. (Optional) To get a copy of the connection string that is created with this window, click *Tools* > *Copy Connection String to Clipboard*.
5. Click *Connect*. SQL Central connects to the database.
6. (Optional) Query the Customers table from Interactive SQL.

a. Open Interactive SQL. In SQL Central, click *File* > *Open Interactive SQL*.
b. In the Interactive SQL window, type the following statement:

```
SELECT * FROM Customers;
```

c. Execute the statement. Press F5 or click *SQL* > *Execute*.

   The result set appears in the *Results* pane.

Results

SQL Central is started and connected to the sample database.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Connecting to the Sample Database [page 21]

Previous task: Lesson 1: Connecting to the Sample Database (SQL Anywhere 17 Demo) [page 22]

Next task: Lesson 3: Connecting to the Sample Database (SQL) [page 26]

Related Information

The SQL Anywhere Sample Database (demo.db)

1.1.1.5.3 Lesson 3: Connecting to the Sample Database (SQL)

Start Interactive SQL and connect to the sample database by using a command prompt.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Procedure

Run the following command to start Interactive SQL and connect to the sample database:

```
dbisql -c "UID=DBA;PWD=sql;DBF=%SQLANYSAMP17%\demo.db"
```

Results

Interactive SQL starts and connects to the sample database.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Connecting to the Sample Database [page 21]

Previous task: Lesson 2: Connecting to the Database (SQL Central) [page 25]

Next task: Lesson 4: Stopping the Database Server (dbstop utility) [page 27]

Related Information

The SQL Anywhere Sample Database (demo.db)

1.1.1.5.4 Lesson 4: Stopping the Database Server (dbstop utility)

Stop the database server by using the Stop Server utility.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Context

On Windows, you can also stop a database server by clicking *Shut Down* on the database server messages window.

Procedure

1. Run the following command to stop the database server that is running the sample database:

   ```
   dbstop demo17 -c "uid=dba;pwd=sql"
   ```

2. If you are prompted that the database has connections, type **Y**.

Results

The database server is stopped and all connections to the sample database are terminated.

Task overview: Tutorial: Connecting to the Sample Database [page 21]

Previous task: Lesson 3: Connecting to the Sample Database (SQL) [page 26]

Related Information

Database Servers [page 318]
Recreating the Sample Database (demo.db)
Stop Server Utility (dbstop) [page 1219]

1.1.1.6 Tutorial: Creating a SQL Anywhere Database

Use SQL Central to create a simple database that is modeled on the Products, SalesOrderItems, SalesOrders, and Customers tables of the SQL Anywhere sample database.

Prerequisites

In the new database, you must have the CREATE ANY OBJECT system privilege.

1. Lesson 1: Creating a Database File [page 29]
   Create a database file to hold your data.
2. Lesson 2: Adding Tables to the Database [page 30]
   Create tables to store the information in your database.
3. Lesson 3: Setting a NOT NULL Constraint on a Column [page 33]
   Add a NOT NULL constraint to a column to prevent the column from allowing NULL values.
4. Lesson 4: Creating a Foreign Key [page 34]
   Create foreign keys to create relationships between tables.

1.1.1.6.1 Lesson 1: Creating a Database File

Create a database file to hold your data.

Prerequisites

You must have the roles and privileges listed at the beginning of this tutorial.

Context

The database file contains system tables and other system objects that are common to all databases.

Procedure

1. Start SQL Central. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Central*.
2. Start the *Create Database Wizard*. Click *Tools* > *SQL Anywhere 17* > *Create Database*.
   a. On the *Welcome* page, click *Next*.
   b. Click *Create a database on this computer*, and then click *Next*.
   c. In the *Save the main database file to the following file* field, type `c:\temp\mysample.db`.

      If your temporary directory is somewhere other than `c:\temp`, specify the appropriate path.
   d. On the *Specify DBA user and Password* page, create the user ID and password for the DBA user.
   e. Click *Finish* and then click *Close*.

      You are automatically connected to the database as the DBA user.

Results

You have created and connected to a database.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Creating a SQL Anywhere Database [page 28]

Next task: Lesson 2: Adding Tables to the Database [page 30]

Related Information

The SQL Anywhere Sample Database (demo.db)
Creating a Database (dbinit Utility) [page 279]
Creating a Database (SQL Central) [page 278]
CREATE DATABASE Statement

1.1.1.6.2 Lesson 2: Adding Tables to the Database

Create tables to store the information in your database.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Procedure

1. Start the *Create Table Wizard*. In the left pane of SQL Central, right-click *Tables* and click *New* > *Table*.
   a. In the *What do you want to name the new table* field, type `Products`.
   b. Click *Finish*.

The database server creates the table using defaults, and then displays the *Columns* tab in the right pane. The *Name* field for the new column is selected and you are prompted to specify a name for the new column.

2. Type **`ProductID`** as the name for the new column.

   Since this is the first column in the table, *PKey* is selected, indicating that the column is the primary key for the table.
3. In the *Data Type* list, click *Integer*.
4. Click the ... button.
5. Click the *Value* tab and click *Default value* > *System-defined* > *autoincrement*.

   An AUTOINCREMENT value increments for each row added to the table. This value ensures that values in the column are unique, which is a requirement for primary keys.
6. Click *OK*.
7. Click *File* > *New* > *Column*.
   a. In the *Name* field, type **`ProductName`**.
   b. In the *Data Type* list, click *Char*.
   c. In the *Size* list, click *15*.
8. Repeat the previous steps to add the following tables to your database:

   Customers table

   Add a table named **`Customers`** with the following columns:

   CustomerID

   The identification number of each customer. Ensure that *PKey* is selected, set the *Data Type* to *Integer*, and set the *Default value* to *autoincrement*.

   CompanyName

   The name of each company. Set the *DataType* to *Char* with a maximum length of 35 characters.

   SalesOrders table

   Add a table named **`SalesOrders`** with the following columns:

   SalesOrderID

   The identification number of each sales order. Ensure that *PKey* is selected, set the *Data Type* to *Integer*, and set the *Default value* to *autoincrement*.

   OrderDate

   The date on which the order was placed. Set the *DataType* to *Date*.

   CustomerID

   The identification number of the customer who placed the sales order. Set the *DataType* to *Integer*.

   SalesOrderItems table

   Add a table named **`SalesOrderItems`** with the following columns:

   SalesOrderItemsID

   The identification number of the sales order that the item belongs to. Ensure that *PKey* is selected and set the *Data Type* to *Integer*.

   LineID

The identification number of each sales order. Ensure that *PKey* is selected and set the *Data Type* to *Integer*.

Since *PKey* is set for both SalesOrderItemsID and LineID, the primary key for the table comprises the concatenated values of these two columns.

ProductID

The identification number of the product being ordered. Set the *DataType* to *Integer*.

9. Click *File* > *Save*.

Results

You have created four tables for your database.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Creating a SQL Anywhere Database [page 28]

Previous task: Lesson 1: Creating a Database File [page 29]

Next task: Lesson 3: Setting a NOT NULL Constraint on a Column [page 33]

Related Information

Primary Keys
Creating a Table
Viewing Entity-relationship (ER) Diagrams [page 1108]

1.1.1.6.3 Lesson 3: Setting a NOT NULL Constraint on a Column

Add a NOT NULL constraint to a column to prevent the column from allowing NULL values.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Context

By default, columns allow NULLs, but it is good practice to declare columns NOT NULL unless there is a reason to allow NULLs.

Procedure

1. In the left pane of SQL Central, double-click *Tables*.
2. Click *Products*, and then click the *Columns* tab in the right pane.
3. Click the *ProductName* column.
4. Click *File* > *Properties*.
5. Click the *Constraints* tab and click *Values cannot be null*.
6. Click *OK*.
7. Click *File* > *Save*.

Results

This constraint means that for each row added to the *Products* table, the *ProductName* column must have a value.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Creating a SQL Anywhere Database [page 28]

Previous task: Lesson 2: Adding Tables to the Database [page 30]

Next task: Lesson 4: Creating a Foreign Key [page 34]

Related Information

Constraint Considerations [page 318]
NULL Special Value

1.1.1.6.4 Lesson 4: Creating a Foreign Key

Create foreign keys to create relationships between tables.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Procedure

1. In the left pane of SQL Central, double-click *Tables*.
2. Click the *SalesOrderItems* table.
3. In the right pane, click the *Constraints* tab.
4. Start the *Create Foreign Key Wizard*. Click *File* > *New* > *Foreign Key*.
   a. In the *To which table do you want this foreign key to refer* list, click the *Products* table.
   b. In the *What do you want to name the new foreign key* field, type `ProductIDkey`.
   c. Click *Next*.
   d. In the *Do you want this foreign key to reference the primary key or a unique constraint* field, click *Primary key*.
   e. In the *Foreign Column* list, click *SalesOrderItemsID*.
   f. Click *Finish*.

Results

You have created a foreign key for the SalesOrderItems table.

You have completed the tutorial on creating a database.

Next Steps

Insert information into your database.

Task overview: Tutorial: Creating a SQL Anywhere Database [page 28]

Previous task: Lesson 3: Setting a NOT NULL Constraint on a Column [page 33]

Related Information

Foreign Keys
Creating a Foreign Key (SQL Central)
Creating a Foreign Key (SQL)

1.1.2 Connection Parameters and Connection Strings

When connecting to a database, the client application uses a set of **connection parameters** and **connection strings** to define the connection.

Connection parameters specify information such as the database server name, the database name, and the user ID.

Each connection parameter specifies a keyword-value pair of the form `parameter=value`. The following example specifies the password connection parameter PWD:

```
PWD=passwd
```

Connection Parameters Passed as Connection Strings

Connection parameters are assembled into **connection strings**. In a connection string, a semicolon separates each connection parameter:

```
parameter1=value1;parameter2=value2;...
```

For example:

```
UID=DBA;PWD=passwd;Host=myhost;DatabaseName=mydemo;ServerName=myserver
```

Generally, the connection string that is built by an application and passed to the interface library does not correspond directly to the way users enter information. Instead, a user may fill in multiple fields in a connection window, or the application may read connection information from an initialization file.

Many utilities accept a connection string as the -c option and pass the connection string unchanged to the interface library. The following example is a typical Backup utility (dbbackup) command line:

```
dbbackup -c "HOST=myhost-pc;DBN=mydemo;UID=DBA;PWD=passwd;ServerName=myserver"
SQLAnybackup
```

In your application, you *must* enter a connection string with the parameter settings separated by semicolons.

In this section:

Related Information

Alphabetical List of Connection Parameters [page 45]
Network Protocol Options [page 187]
Character Set Conversion [page 606]

1.1.2.1 Common Connection Scenarios

There are several common connection scenarios that help in understanding how to effectively connect to a database.

The connection parameters required to connect to a database vary depending on whether multiple database servers are running on the same computer, whether multiple databases are running on the same database server, and whether the client application is running on the same computer as the database server.

> **i Note**
>
> A single computer can run multiple database servers at the same time. For this reason, it is recommended that you always specify the database server name (ServerName=`server-name`) when authenticating to a database server.
>
> Each database server can run multiple databases at the same time. For this reason, it is recommended that you always specify the database name (DatabaseName=`database-name`) when authenticating to a database server.

The following tables contain several common connection scenarios and the client connection parameters that are required in each case. By combining these connection parameters, you can correctly identify the database you want to connect to. In addition to those listed below, many connection parameters for handling less common connection scenarios are supported.

Shared Memory Connections to Local Servers

An X in the following table indicates a required connection parameter for the scenario listed at the top of the column.

Connection parameter	1 server hosting 1 database	1 server hosting many databases	Many servers each hosting 1 database	Many servers each hosting many databases
UserID (UID)	X	X	X	X
Password (PWD)	X	X	X	X
DatabaseName (DBN)		X		X
ServerName(Server)	[1]	[1]	X	X
Host [2]	[1]	[1]		

[1] Include the server name (ServerName=`server-name`) if the server is running with the -xd option.

[2] When the Host connection parameter (Host=`host-name`:`port-number`) is used, the connection is attempted using TCP/IP rather than shared memory. This option is inappropriate for the personal database server unless the TCP/IP communication protocol was enabled on the personal database server using the -x option. Also, you must use the host name **`localhost`** when connecting to a personal database server using TCP/IP.

Connections to Network Servers

An X in the following table indicates a required connection parameter for the scenario listed at the top of the column.

Connection parameter	1 server hosting 1 database	1 server hosting many databases	Many servers each hosting 1 database	Many servers each hosting many databases
UserID (UID)	X	X	X	X
Password (PWD)	X	X	X	X
DatabaseName (DBN)		X		X
ServerName(Server)	[3]	[3]	X	X
Host	X[3]	X[3]	X	X

[3] Include the port number (Host=`host-name:port-number`) or the server name (ServerName=`server-name`) if the server is not running on the default TCP/IP port (2638). If the database server with the non-default port number is on macOS or if it was started using the -sb option then you must specify the port number (Host=`host-name:port-number`) to connect.

Example

Only one database server running only one database, running on the same computer as the client application

```
dbisql -c "UID=DBA;PWD=sql"
```

Only one database server running multiple databases, running on the same computer as the client application

```
dbisql -c "UID=DBA;PWD=sql;DBN=demo"
```

Multiple database servers each running one database, running on the same computer as the client application

```
dbisql -c "UID=DBA;PWD=sql;ServerName=demo17"
```

Multiple database servers each running several databases, running on the same computer as the client application

```
dbisql -c "UID=DBA;PWD=sql;ServerName=demo17;DBN=demo"
```

Only one database server running only one database, running on a different computer than the client application, and the computer running the database server is named myhost-pc

```
dbisql -c "UID=DBA;PWD=sql;Host=myhost-pc"
```

Multiple database servers each running several databases, running on a different computer than the client application, and the computer running the database server is named myhost-pc

```
dbisql -c "UID=DBA;PWD=sql;Host=myhost-pc;ServerName=demo17;DBN=demo"
```

You do not know if the database server or database are running and you want them started on the same computer as the client application so you can connect to it

```
dbisql -c "UID=DBA;PWD=sql;ServerName=demo17;DBF=%SQLANYSAMP17%\demo.db"
```

In this section:

Related Information

Alphabetical List of Connection Parameters [page 45]
Userid (UID) Connection Parameter [page 115]
Password (PWD) Connection Parameter [page 104]
DatabaseName (DBN) Connection Parameter [page 71]
ServerName (Server) Connection Parameter [page 110]
Host Connection Parameter [page 86]
-xd Database Server Option [page 526]

1.1.2.1.1 Default Connection Parameters

You can use default behavior to make a connection and leave some of the connection parameters unspecified.

However, using the default behavior in a production environment can cause problems if the application is installed with other SQL Anywhere applications. If you are having problems connecting, explicitly specifying values instead of relying on defaults may resolve your problem.

It is recommended that you always specify the server name using the ServerName connection parameter especially if you are running an embedded SQL Anywhere server.

No Defaults for a Local Server (Recommended Method)

Use the following connection string to connect to a named local server, using a named database:

```
ServerName=server-name;DBN=db-name;UID=user-id;PWD=password
```

Default Database Server and Database

Use the default parameters to connect when you are sure that there is only one database server running on the local computer:

```
UID=user-id;PWD=password
```

Default Database Server

When you are sure that only one database server is running on the local computer, you can use the default server settings and specify the database you want to connect to:

```
DBN=db-name;UID=user-id;PWD=password
```

Default Database

If more than one database server is running on the local computer, specify which server you want to connect to. You do not need to specify the database name if only a single database is running on that database server. The following connection string connects to a named server, using the default database:

```
ServerName=server-name;UID=user-id;PWD=password
```

Default Database Server on Network Computer

The following connection string connects to the specified database running on the default database server that is running on a network computer. This connection string assumes that the database server is running on the default TCP/IP port 2638.

```
Host=host-name;DBN=db-name;UID=user-id;PWD=password
```

If the database server is not using the default TCP/IP port (2638), then you must specify the database server name (ServerName=`server-name`) or the port number (Host=`host-name:port-number`). If you specify the server name, you may not be able to connect if the server is on macOS or if the database server was started using the -sb option. In these cases, you must specify the port number (Host=`host-name:port-number`) to connect.

Related Information

ServerName (Server) Connection Parameter [page 110]

1.1.2.2 Connection Parameter Syntax Rules

There are several rules governing connection parameter syntax.

Connection strings containing spaces

When specifying a connection string on the command line, you must enclose the entire connection string in double quotes if any of the connection parameter values contains spaces.

Boolean values

Boolean (true or false) arguments are either YES, ON, 1, TRUE, Y, or T if true, or NO, OFF, 0, FALSE, N, or F if false.

Case sensitivity

Connection parameter names are case insensitive, although their values may not be (for example, file names on UNIX or Linux).

Character set restrictions

It is recommended that the database server name (specified by the ServerName connection parameter) be composed of the ASCII character set in the range 32 to 126. This limitation does not apply to other connection parameter values.

Priority

> ⚠ Caution
>
> Do not specify a parameter more than once in a connection string. If you specify a parameter more than once, the results are not guaranteed and may be inconsistent. As well, the behavior may change in a future release of the software.

The following rules govern the priority of connection parameters:

- The entries in a connection string are read left to right. If the same connection parameter is specified more than once, the last one in the string applies. ODBC and OLE DB client applications (except for SQL Central and Interactive SQL) are exceptions to this behavior: if the same parameter is specified more than once, then the first value in the string applies.
- In order of precedence, the interface library gets connection parameters from the following sources:

  Connection string

  You can pass parameters explicitly in the connection string.

  SQLCONNECT environment variable

  The SQLCONNECT environment variable can store connection parameters.

  Data sources

  ODBC data sources can store connection parameters.

  If a connection string and an ODBC data source both specify the same connection parameter, then the value from the connection string is used and the value from the data source is ignored.

  For example, assume that the database server demo17 is running the databases demo and demo2. The following command creates a data source for the demo database.

  ```
  dbdsn -w demodsn -c "ServerName=demo17;Host=myhost-pc;DBN=demo" -y
  ```

  The connection string DBN value (DBN=demo2) has higher precedence than the data source's DBN value.

  ```
  dbping -d -c "UID=DBA;PWD=sql;DBN=demo2;DSN=demodsn"
  ```

  The connection parameters used to connect are:

  ```
  UID=DBA;PWD=sql;DBN=demo2;ServerName=demo17;Host=myhost-pc
  ```

Connection string parsing

If there is a problem parsing the connection string, an error is generated that indicates which connection parameter caused the problem.

Empty connection parameters

Connection parameters that are specified with empty values are treated as a zero length string.

Connection string maximum length The maximum connection string length is 8 K.

Related Information

Alphabetical List of Connection Parameters [page 45]
Connection Strings and Character Sets [page 607]

1.1.2.3 Connection Parameters to Use with Embedded Databases

An **embedded database** is typically designed for use by a single application, runs on the same computer as the application, and is hidden from the user.

When an application uses an embedded database, the personal database server is generally not running when the application connects. The database is started using a connection string, and by specifying the database file in the DatabaseFile (DBF) parameter of the connection string.

To improve query performance for databases that are started automatically, start the database as soon as possible, even if users are not connecting right away. This allows the cache to warm before queries are executed against the database.

The ServerName (Server) Connection Parameter

When using an embedded database, use the ServerName (Server) connection parameter. This ensures that the database connects to the correct database server if there are other applications running SQL Anywhere database servers on the same computer.

If the database server has an alternate server name, the client application can only use the alternate server name to connect to the database that specified the alternate server name. You cannot use the alternate server name to connect to any other databases running on that database server.

The DatabaseFile (DBF) Connection Parameter

The DBF connection parameter specifies the database file to use. The database file automatically starts on the database server matching the ServerName connection parameter (if it is not already running) and a database server is automatically started (if it is not already running).

The database automatically stops when there are no more connections to the database (generally when the application that started the connection disconnects). If the database server was automatically started, the database server stops once the database stops.

In the following example, the sample database is started as an embedded database:

```
"DBF=%SQLANYSAMP17%\demo.db"
```

Using the StartLine [START] Connection Parameter

The following connection parameters show you how to customize the startup of the sample database as an embedded database. The StartLine connection parameter is useful for using database server options. For example:

```
"START=dbeng17 -xd -c 8M;DBF=%SQLANYSAMP17%\demo.db"
```

Specifying the -xd option prevents the database server from becoming the default database server.

To prevent users from starting or stopping the embedded database, or starting other databases on the database server, set the -gd option to **none**.

There are many connection parameters that affect how a database server is started. It is recommended that you use the following connection parameters instead of providing the corresponding server options within the StartLine (START) connection parameter:

- ServerName (Server)
- DatabaseFile (DBF)
- DatabaseSwitches (DBS)
- DatabaseName (DBN)

Using the Elevate Connection Parameter

Specifying ELEVATE=YES in your connection string setting allows non-elevated client processes to start elevated servers automatically, which may be necessary if administrative rights are required to access the database file. This parameter is ignored if the database server is not started automatically.

Example

The following is an example of a complete connection string for an application using an embedded database:

```
"UID=DBA;PWD=passwd;DBF=c:\myapp\mydb.db;ServerName=myapp;START=dbeng17 -c 12M -
xd"
```

Related Information

Cache Warming [page 1445]
Connection IDs [page 256]
DatabaseFile (DBF) Connection Parameter [page 68]
ServerName (Server) Connection Parameter [page 110]
StartLine (START) Connection Parameter [page 112]
-sn Database Option [page 560]
-gd Database Server Option [page 430]
-xd Database Server Option [page 526]
Elevate Connection Parameter [page 77]

1.1.2.4 How the Database Utilities Obtain Their Connection Parameters

All database utilities use Embedded SQL to communicate with the database server.

Many of the administration utilities obtain the connection parameter values by:

1. Using values specified on the command line. For example, the following command starts a backup of the MyTestDB database on the default server using the user ID DBA and the password passwd:

   ```
   dbbackup -c "UID=DBA;PWD=passwd;DBN=MyTestDB" c:\backup
   ```

2. Using the SQLCONNECT environment variable settings if any values are missing. SQL Anywhere does not set this variable automatically.

Related Information

Database Administration Utilities [page 1115]
SQLCONNECT Environment Variable [page 582]

1.1.2.5 Connection Parameters in an ODBC Data Source

You can use an ODBC data source to connect to a database.

The ODBC data source resides on the client computer.

The ODBC data source contains a set of connection parameters. You can store sets of connection parameters as an ODBC data source, in either the Windows Registry or as files.

For SQL Anywhere, ODBC data sources can be used for more than just ODBC applications. ODBC data sources can be used by all supported client interfaces except SAP Open Client and jConnect.

If you have a data source, your connection string can name the data source to use:

Data source name

Use the DataSourceName (DSN) connection parameter to reference a data source in the Windows Registry:

```
DSN=my-data-source
```

File data source

Use the FileDataSourceName (FILEDSN) connection parameter to reference a data source held in a file:

```
FileDSN=mysource.dsn
```

> **i** Note
>
> If a connection string and an ODBC data source both specify the same connection parameter, the value from the connection string is used and the value from the data source is ignored.

Related Information

File Data Sources on Microsoft Windows [page 124]
Alphabetical List of Connection Parameters [page 45]
Connection Parameter Syntax Rules [page 40]

1.1.3 Alphabetical List of Connection Parameters

Connection parameters are included in connection strings.

They can be entered in the following places:

- In an application's connection string.
- SQLConnect environment variable.
- In an ODBC data source.
- In the SQL Anywhere *Connect* window.

> **i** Note
>
> - Connection parameters are case insensitive, although their values may not be (for example, file names on UNIX and Linux).
> - Boolean parameters are turned on with YES, Y, ON, TRUE, T, or 1, and are turned off with any of NO, N, OFF, FALSE, F, and 0. The parameter values are case insensitive (for example, YES, Yes, yes. etc. are equivalent).
> - The Usage for each connection parameter describes the circumstances under which the parameter is to be used. Common usage entries include the following:
>
> **Embedded databases**

The personal database server (dbeng17) is typically used with embedded applications. When the application disconnects from the database, the database is unloaded and the server stops.

Local database servers

A local database server refers to a server running on your personal computer. You can use either the personal server (dbeng17) or the network server (dbsrv17).

Network database servers

A network server (dbsrv17) can be run on your personal computer or on another computer on a network. Use the network server when you want others to be able to connect to the database.

- You can use the dbping utility to test connection strings. The -c option is used to specify the connection parameters. For example, suppose a database server with the name demo17 is running the sample database (which can be started with the command `dbsrv17 -n demo17 "%SQLANYSAMP17%\demo.db"`). The following command returns a message indicating that the connection to the database was successful.

  ```
  dbping -d -c "Server=demo17;DBN=demo;UID=DBA;PWD=sql"
  ```

  The following command returns the message `Ping server failed - Database server not found` if no database server named other-server is running on the local computer:

  ```
  dbping -c "Server=other-server"
  ```

In this section:

Language (LANG) Connection Parameter [page 93]
Specifies the language of the connection.

LazyClose (LCLOSE) Connection Parameter [page 94]
Controls whether cursor requests are queued until the next request or performed immediately.

LivenessTimeout (LTO) Connection Parameter [page 95]
Controls the shutdown of connections when they are no longer intact.

LogFile (LOG) Connection Parameter [page 97]
Sends client error messages and debugging messages to a file.

NewPassword (NEWPWD) Connection Parameter [page 98]
Allows users to change passwords, even if they have expired, without DBA intervention.

MatView (MATVIEW) Connection Parameter [page 100]
Specifies the string to return for materialized views when the ODBC function SQLTables is called.

NodeType (NODE) Connection Parameter [page 101]
Selects which type of server, participating in a mirroring configuration, to connect to.

Password (PWD) Connection Parameter [page 104]
Provides a password for a connection.

PrefetchBuffer (PBUF) Connection Parameter [page 105]
Sets the maximum amount of memory for buffering rows, in bytes.

PrefetchOnOpen Connection Parameter [page 107]
Sends a prefetch request with a cursor open request when this parameter is enabled.

PrefetchRows (PROWS) Connection Parameter [page 107]
Provides an initial suggested number of rows to prefetch when querying the database.

RetryConnectionTimeout (RetryConnTO) Connection Parameter [page 109]
Instructs the client library (DBLIB, ODBC, ADO, and so on) to keep retrying the connection attempt, as long as the server is not found, for the specified period of time.

ServerName (Server) Connection Parameter [page 110]
Specifies the name of a running database server to which you want to connect.

StartLine (START) Connection Parameter [page 112]
Starts a local database server running from an application.

Unconditional (UNC) Connection Parameter [page 114]
Stops a database server using the db_stop_engine function, or a database using the db_stop_database function, even when there are connections to the database server.

Userid (UID) Connection Parameter [page 115]
Specifies the user ID used to log in to the database.

Related Information

Connection Parameters and Connection Strings [page 35]
Connection Parameter Syntax Rules [page 40]
ODBC Data Sources [page 116]
Ping Utility (dbping) [page 1195]

Troubleshooting: How Connections Are Established [page 262]

1.1.3.1 AppInfo (APP) Connection Parameter

Assists administrators in identifying the origin of particular client connections from a database server.

Syntax

```
{ AppInfo | APP }=keyword=value
```

Usage

Anywhere

Allowed Values

Clients can specify their own string that is appended to the generated string. The AppInfo property string is a sequence of semicolon-delimited `key=value` pairs. The valid keys are as follows:

API

DBLIB, ODBC, OLEDB, ADO.NET, iAnywhereJDBC, PHP, PerlDBD, or DBEXPRESS.

APPINFO

If you specified AppInfo in the connection string, the string entered.

EXE

The name of the client executable.

HOST

The host name of the client computer.

IP

The IP address of the client computer.

OS

The operating system name and version number.

OSUSER

The operating system user name associated with the client process. If the client process is impersonating another user (or the set ID bit is set on UNIX or Linux), the impersonated user name is returned. An empty string is returned for version 10.0.1 and earlier clients, and for HTTP and TDS clients.

PID

The process ID of the client.

THREAD

The thread ID of the client.

TIMEZONEADJUSTMENT

The number of minutes that must be added to the Coordinated Universal Time (UTC) to display time local to the connection.

VERSION

The version of the client library in use, including major and minor values, and a build number (for example 17.0.11.1293).

Default

Empty string

Remarks

This connection parameter is sent to the database server from Embedded SQL, ODBC, OLE DB, or ADO.NET clients and from applications using the JDBC driver. It is not available from SAP Open Client or jConnect applications.

It consists of a generated string that holds information about the client process, such as the IP address of the client computer, the operating system it is running on, and so on. The string is associated in the database server with the connection, and you can retrieve it using the following statement:

```
SELECT CONNECTION_PROPERTY( 'AppInfo' );
```

If you specify a debug log file in your client connection parameters, the AppInfo string is added to the file.

Example

Connect to the sample database from Interactive SQL (the JDBC driver is used by default):

```
dbisql -c "UID=DBA;PWD=sql;DBF=%SQLANYSAMP17%\demo.db"
```

In Interactive SQL, view the application information:

```
SELECT CONNECTION_PROPERTY( 'AppInfo' );
```

The result is as follows (in a single string):

```
IP=ip-address;
HOST=computer-name;
OSUSER=user-name;
OS='operating-system-name';
EXE='C:\Program Files\SQL Anywhere 17\Bin64\dbisql.exe';
PID=0x1170;
THREAD=0x24a4;
```

```
VERSION=17.0.11.1293;
API=iAnywhereJDBC;
TIMEZONEADJUSTMENT=-240
```

Connect to the sample database from Interactive SQL, appending your own information to the AppInfo property:

```
dbisql -c "UID=DBA;PWD=sql;DBF=%SQLANYSAMP17%\demo.db;APP=Interactive SQL
connection"
```

View the application information:

```
SELECT CONNECTION_PROPERTY( 'AppInfo' );
```

The result is as follows (in a single string). Note the appearance of the APPINFO value you specified at the end:

```
IP=ip-address;
HOST=computer-name;
OSUSER=user-name;
OS='operating-system-name';
EXE='C:\Program Files\SQL Anywhere 17\Bin64\dbisql.exe';
PID=0x1170;
THREAD=0x24a4;
VERSION=17.0.11.1293;
API=iAnywhereJDBC;
TIMEZONEADJUSTMENT=-240;
APPINFO='Interactive SQL connection'
```

Related Information

Connection Parameters and Connection Strings [page 35]
request_timeout Option [page 810]

1.1.3.2 AppConnTimeout Connection Parameter

Specifies how long to wait for the first reply from the server during the connection attempt before the connection times out.

Syntax

```
{ AppConnTimeout }=timeout-value
```

Usage

Anywhere

Allowed Values

timeout-value

The timeout period for connections, in seconds. The minimum value is 1 second. If you specify 0, then the timeout value is 300 seconds for shared memory connections and is 30 seconds for TCP connections.

Default

0

Remarks

This connection parameter sets a timeout value for both shared memory connections and TCP connections.

This timeout does not include the initial time it takes to locate a server over TCP/IP and the time to wait for a response when establishing TCP/IP communications (specified by the TCP Timeout (TO) protocol option). AppConnTimeout kicks in after this interval and a timeout that occurs here may be indicative of an overloaded database server.

When you specify a value other than 0 for AppConnTimeout, the timeout value applies to both shared memory and TCP connections.

Once a TCP connection is made, then the LivenessTimeout (LTO) connection parameter setting applies. There is no liveness timeout value for shared memory connections.

Related Information

LivenessTimeout (LTO) Connection Parameter [page 95]

1.1.3.3 AutoStart (ASTART) Connection Parameter

Controls whether a personal database server is started if no database server can be connected to.

Syntax

```
{ AutoStart | ASTART }={ YES | NO }
```

Usage

Anywhere

Default

YES

Remarks

By default, if no database server is found during a connection attempt, and a database file, database name, or the START connection parameter is specified, then a personal database server is started on the same computer. You can turn this behavior off by setting the AutoStart (ASTART) connection parameter to NO in the connection string. The database server is not started automatically if the Host connection parameter is specified or the CommLinks (LINKS) parameter includes TCPIP.

To improve query performance for databases that are started automatically, start the database as soon as possible, even if users are not connecting immediately. Doing so allows the cache to warm before queries are executed against the database.

> **i** Note
>
> The AutoStart connection parameter controls whether or not a database server is automatically started. It does *not* control whether a database is automatically started. Use the DatabaseName connection parameter to connect to a database that is already started. Use the DatabaseFile connection parameter to connect to a database that is already started or to automatically start and connect to a database if necessary.

Related Information

Connection Parameters and Connection Strings [page 35]
Cache Warming [page 1445]
DatabaseName (DBN) Connection Parameter [page 71]
DatabaseFile (DBF) Connection Parameter [page 68]
Troubleshooting: How Database Servers Are Located [page 263]
CommLinks (LINKS) Connection Parameter [page 59]
Elevate Connection Parameter [page 77]
AutoStop (ASTOP) Connection Parameter [page 54]

1.1.3.4 AutoStop (ASTOP) Connection Parameter

Controls whether a database is stopped when there are no more open non-HTTP connections.

Syntax

```
{ AutoStop | ASTOP }={ YES | NO }
```

Usage

Anywhere

Default

YES

Remarks

By default, any database server that is started from a connection string is stopped when there are no more non-HTTP connections to it. As well, any database that is loaded from a connection string is unloaded when there are no more non-HTTP connections to it. This behavior is equivalent to AutoStop=YES.

If you supply AutoStop=NO, any database that you start in that connection remains running when there are no more non-HTTP connections to it. As a result, the database server remains operational as well.

If the only connection to a database is an HTTP connection, and the database is configured to stop automatically, when the HTTP connection disconnects, the database does not stop automatically. As well, if a database that is configured to stop automatically has an HTTP connection and a command sequence or TDS connection, when the last command sequence or TDS connection disconnects, the database stops, and any HTTP connections are dropped.

The AutoStop (ASTOP) connection parameter is used only if you are connecting to a database that is not currently running. It is ignored if the database is already started.

In .NET applications, you should be careful when using the AutoStop connection parameter. Closing a connection will close it as far as the application is concerned, but active connections remain open when connection pooling is enabled. As a result the server does not shut down, even though you may expect it to do so.

Related Information

Connection Parameters and Connection Strings [page 35]
.NET Connection Pooling
Database Starting and Stopping [page 328]
-ga Database Server Option [page 427]
AutoStart (ASTART) Connection Parameter [page 52]
START DATABASE Statement

1.1.3.5 CharSet (CS) Connection Parameter

Specifies the character set encoding to be used on this connection.

Syntax

```
{ CharSet | CS }={ NONE | character-set }
```

Usage

Anywhere

Allowed Values

NONE

Specifying CharSet=NONE requests that the connection use the database CHAR character set encoding. This is the default behavior.

character-set

Specify the label for the character set encoding that you want to use.

Default

DBLIB (ESQL,Perl, PHP, Python, Ruby), ODBC in ANSI mode: The local (client) operating system character set encoding.

ADO.NET, JDBC, ODBC in UNICODE mode, OLE DB: The database CHAR character set encoding.

JavaScript: The database NCHAR character set encoding.

Remarks

If you supply a value for CharSet, the specified character set encoding is used for the current connection. Conversions are attempted from the database CHAR character set encoding to the specified connection character set encoding. The NCHAR character set encoding is not affected by this connection parameter.

ADO.NET, JDBC, and OLE DB are Unicode application programming interfaces so the CharSet setting is ignored.

The character set encoding label that you specify may be an alternate for the preferred label. In this case, the preferred label is reported by the CharSet connection property. For example, the following Interactive SQL command specifies the `cp1251` character set encoding in the connection string.

```
dbisql -c "UID=DBA;PWD=passwd;CharSet=cp1251" "SELECT
connection_property('CharSet')"
```

The result returned is:

```
connection_property('charset')
------------------------------
windows-1251
```

Some alternate character set encoding labels are not unique and should be avoided (for example, `ibm-943_P15A-2003`).

When unloading data, you can specify the character set using the CharSet connection parameter.

To avoid lossy character set conversions, setting the CharSet connection parameter is not recommended when using Unicode client APIs. Unicode client APIs include ADO.NET, OLE DB, and the SQL Anywhere JDBC driver. ODBC is also a Unicode client API when the wide (Unicode) functions are used.

Example

The following connection string fragment specifies that Windows 1252 character set encoding should be used for the connection:

```
CharSet=windows-1252
```

Related Information

Connection Parameters and Connection Strings [page 35]
Locale Information [page 616]
Locale Character Set [page 615]
SACHARSET Environment Variable [page 574]
Recommended Character Sets and Collations [page 634]

1.1.3.6 ClientFileValidator (CFV) Connection Parameter

Controls how the database server verifies that a request to read or write a local file is coming from a trusted source.

Syntax

```
{ ClientFileValidator | CFV } = { TrustedServer | Callback | Never }
```

Usage

Anywhere

Allowed Values

Never

Never allow file transfer.

Callback

As defined in the DB_CALLBACK_VALIDATE_FILE_TRANSFER function. Transfer the file only if a callback is defined and returns TRUE.

TrustedServer

Transfer the file if the database server determines that the request comes directly from the client and is not the result of an indirect request, or if a callback returns TRUE.

Default

TrustedServer

Remarks

This connection parameter takes precedence over all other options that also control files.

Example

The following example prohibits all file transfers from the client to the database server:

```
CFV=Never
```

1.1.3.7 CommBufferSize (CBSIZE) Connection Parameter

Sets the maximum size of communication packets.

Syntax

```
{ CommBufferSize | CBSIZE }=size[ k ]
```

Usage

Anywhere

Allowed Values

size

This integer specifies the maximum size of communication packets. The default value is in bytes. Use k to specify units of kilobytes. The minimum value of CommBufferSize is 1000 bytes, and the maximum is 65535 bytes.

Default

If no CommBufferSize value is set, then the CommBufferSize is controlled by the setting on the database server, which defaults to 7300 bytes.

The default CommBufferSize for connections between mirror servers is 64240 bytes.

Remarks

The protocol stack sets the maximum size of a packet on a network. If you set the CommBufferSize to be larger than that permitted by your network, then the communication packets are broken up by the network software. The default size is a multiple of the standard ethernet TCP/IP maximum packet size (1460 bytes).

A larger packet size may improve performance for multi-row fetches and fetches of larger rows, but it also increases memory usage for both the client and the database server.

If CommBufferSize is not specified on the client, then the connection uses the database server's buffer size. If CommBufferSize is specified on the client, then the connection uses the CommBufferSize value.

Using the -p database server option to set the CommBufferSize causes all clients that do not specify their own CommBufferSize to use the size specified by the -p database server option.

Example

Set the buffer size to 1460 bytes:

```
...
CommBufferSize=1460
...
```

Alternatively, you can set this parameter by entering its value in the *CommBufferSize* text box on the *Advanced* tab of the *ODBC Configuration For SQL Anywhere* window.

Related Information

Connection Parameters and Connection Strings [page 35]
-p Database Server Option [page 477]

1.1.3.8 CommLinks (LINKS) Connection Parameter

Specifies client-side network protocol options.

Syntax

```
{ CommLinks | LINKS }={ [ SharedMemory | ShMem ] | ALL | [ TCPIP | TCP ] }
[, ... ] string
```

Usage

The CommLinks (LINKS) connection parameter can be used for connections to a network database server (dbsrv17) and is optional for connections to a personal database server (dbeng17).

Allowed Values

CommLinks (LINKS) connection parameter values are case insensitive, and include:

SharedMemory (ShMem)

Connect using the shared memory protocol for same-computer communication. This is the default setting. The client tries shared memory first if it is included in a list of protocols specified by the CommLinks(LINKS) connection parameter, regardless of the order in which protocols appear.

ALL

Attempt to connect using the shared memory protocol first, followed by TCP/IP.

TCPIP (TCP)

Connect using the TCP/IP communication protocol. TCP/IP is supported on all operating systems.

You can specify additional network protocol options for TCPIP.

Default

SharedMemory

Remarks

> **i** Note
>
> It is recommended that you only use the CommLinks (LINKS) connection parameter if you need to specify TCP/IP protocol options other than HOST or ServerPort (PORT). Otherwise, use the Host connection parameter.
>
> You cannot specify both CommLinks and Host in a connection string.
>
> When you use the CommLinks (LINKS) connection parameter for a TCP/IP connection, a server name should be specified using the ServerName (Server) connection parameter.

If you do not specify a Host connection parameter or a CommLinks (LINKS) connection parameter, the client searches for a database server on the current computer only, and only using a shared memory connection. This is the default behavior, and is equivalent to CommLinks=ShMem. The shared memory protocol is the fastest communication link between a client and database server running on the same computer, as is typical for applications connecting to a personal database server.

If you specify both TCPIP and SharedMemory or CommLinks=ALL, then shared memory is attempted first, followed by TCP/IP if the database server cannot be found over shared memory. For example, in the following connection string, shared memory is attempted first:

```
"UID=DBA;PWD=sql;Server=demo;DBN=demo;LINKS=TCPIP,SHMEM"
```

A personal database server is not started automatically if the Host connection parameter is specified or the CommLinks (LINKS) parameter includes TCP/IP.

The TCP/IP communication protocol is enabled on the database server using the -x option.

Example

The following connection string fragment attempts to connect to the database server named demo running on the current subnet:

```
CommLinks=tcpip;Server=demo;UID=DBA;PWD=sql;
```

The following connection string fragment attempts to connect to the database server named demo. It first attempts to connect over shared memory. If the database server cannot be located using shared memory, it attempts to connect using TCP/IP.

```
CommLinks=tcpip,shmem;Server=demo;UID=DBA;PWD=sql
```

The following connection string attempts to connect to the server named demo running on the host named kangaroo.

```
CommLinks=tcpip(HOST=kangaroo);Server=demo;UID=DBA;PWD=sql
```

Alternatively, you could use the Host connection parameter to connect to the server named demo running on the host named kangaroo.

```
Host=kangaroo;Server=demo;UID=DBA;PWD=sql
```

Related Information

Network Protocol Options [page 187]
Communication Protocols [page 173]
Connection Parameters and Connection Strings [page 35]
General Security Tips [page 1680]
Host Connection Parameter [page 86]
ServerName (Server) Connection Parameter [page 110]
-x Database Server Option [page 523]
Troubleshooting: How Database Server Address Information Is Cached in sasrv.ini for Faster Connections [page 273]

1.1.3.9 Compress (COMP) Connection Parameter

Turns compression on or off for a connection.

Syntax

```
{ Compress | COMP }={ YES | NO }
```

Usage

Anywhere except with TDS connections. TDS connections (including jConnect) do not support SQL Anywhere communication compression.

Default

NO

Remarks

The packets sent between a client and database server can be compressed using the Compress (COMP) connection parameter. Compressing a connection may improve performance under some circumstances. Large data transfers with highly compressible data tend to get the best compression rates.

If a value is not set for the Compress connection parameter, the compression status is controlled by the setting on the database server, which defaults to no compression. If the client and database server settings are different, the client setting applies.

It is recommended that you conduct a performance analysis on the particular network and using the particular application before using communication compression in a production environment.

To enable compression for all remote connections on the database server, use the -pc server option.

Same-computer connections over any communication link do not enable compression, even if the -pc option or COMPRESS=YES parameter is used.

Example

The following connection string fragment turns packet compression on:

```
Compress=YES
```

The following connection string fragment turns packet compression off:

```
Compress=NO
```

Related Information

Connection Parameters and Connection Strings [page 35]
Communication Compression Settings [page 185]
-pc Database Server Option [page 478]

1.1.3.10 CompressionThreshold (COMPTH) Connection Parameter

Increases or decreases the size limit at which packets are compressed.

Syntax

```
{ CompressionThreshold | COMPTH }=size[ k ]
```

Usage

Anywhere except TDS. Only applies to compressed connections.

Allowed Values

size

This integer specifies the size limit at which packets are compressed. The default value is in bytes, but you can use k to specify units of kilobytes. If both the client and database server specify different compression threshold settings, the client setting applies. The minimum supported value is 1 byte, and the maximum supported value is 32767 bytes. Values less than 80 bytes are not recommended.

Default

120

If no CompressionThreshold value is set, the compression threshold value is controlled by the setting on the server, which defaults to 120 bytes.

Remarks

Changing the compression threshold can help performance of a compressed connection by allowing you to only compress packets when compression will increase the speed at which the packets are transferred.

When compression is enabled, individual packets may or may not be compressed, depending on their size. For example, the database server does not compress packets smaller than the compression threshold, even if communication compression is enabled. As well, small packets (less than about 100 bytes) usually do not compress at all. Since CPU time is required to compress packets, attempting to compress small packets could actually decrease performance.

Lowering the compression threshold value may improve performance on very slow networks, while raising the compression threshold may improve performance by reducing CPU. However, since lowering the compression threshold value will increase CPU usage on both the client and server, a performance analysis should be done to determine whether changing the compression threshold is beneficial.

Example

Connect, with a compression threshold of 100 bytes.

```
CompressionThreshold=100
```

Related Information

Communication Compression Settings [page 185]
Connection Parameters and Connection Strings [page 35]
-pt Database Server Option [page 482]

1.1.3.11 ConnectionName (CON) Connection Parameter

Names a connection, to make switching to it easier in multi-connection applications.

Syntax

```
{ ConnectionName | CON }=connection-name
```

Usage

Anywhere

Allowed Values

connection-name

This string specifies a name for the particular connection you are making.

Default

No connection name.

Remarks

This connection parameter is optional. You can leave this value unspecified unless you are going to establish more than one connection, and switch between them.

The connection name is not the same as the data source name.

For a connection to be pooled, the connection name can be different, but all other connection parameters must be identical.

This does not apply to .NET connection pooling. Multiple connection pools are supported by .NET and a different connection name creates a different pool.

The following names are used for temporary connections created by the database server:

- INT:ApplyRecovery
- INT:BackupDB
- INT:Checkpoint
- INT:Cleaner
- INT:CloseDB
- INT:CreateDB
- INT:CreateMirror
- INT:DelayedCommit
- INT:DiagRcvr
- INT:DropDB
- INT:EncryptDB
- INT:Exchange
- INT:FlushMirrorLog
- INT:FlushStats
- INT:HTTPReq
- INT:PromoteMirror
- INT:PurgeSnapshot
- INT:ReconnectMirror
- INT:RecoverMirror
- INT:RedoCheckpoint
- INT:RefreshIndex
- INT:ReloadTrigger
- INT:RenameMirror
- INT:RestoreDB
- INT:StartDB
- INT:VSS

Example

Connect to a database, naming the connection first-con:

```
CON=first-con
```

Related Information

Connection Parameters and Connection Strings [page 35]
SET CONNECTION Statement [Interactive SQL] [ESQL]

1.1.3.12 ConnectionPool (CPOOL) Connection Parameter

Controls the behavior of client connection pooling.

Syntax

```
ConnectionPool={ NO | YES [ ( [ Timeout=timeout-sec; ] [ MaxCached=max-cached-conn ] ) ] }
```

Usage

All platforms except non-threaded Unix clients.

Allowed Values

timeout-sec

The idle timeout period, in seconds, of the connection pool. The default value is 60 seconds. Cached connections that are not reused within the time specified by the `timeout-sec` value are disconnected and are no longer available for reuse.

max-cached-conn

The maximum number of cached connections from each application. The default value is five connections. A connection is cached if it is disconnected and the maximum number of connections specified by the `max-cached-conn` value has not been reached. The connection is reinitialized, and the cached connection remains connected to the database server even though the application has disconnected it.

Default

YES

Remarks

This connection parameter does not apply to Tabular Data Stream (TDS) connections.

For connection pooling in .NET applications, you must use the .NET *Pooling* connection parameter - the ConnectionPool (CPOOL) connection parameter is ignored.

Connection pooling may improve the performance of applications that make multiple, brief connections to the database server. When a connection is disconnected, it is automatically cached and may be reused when the application reconnects.

For a connection to be pooled, the connection name can be different, but all other connection parameter values must be identical. The following two connection strings are considered equivalent for connection pooling.

```
UID=DBA;PWD=passwd;ConnectionName=One
Password=passwd;UserID=DBA;CON=Two
```

This does not apply to .NET connection pooling. Multiple connection pools are supported by .NET and a different connection string creates a different pool. To reuse a pooled connection, the connection strings must be textually identical. For .NET applications, the following two connection strings are not considered equivalent for connection pooling; thus each string would create its own connection pool.

```
UID=DBA;PWD=passwd;ConnectionName=One
UserID=DBA;PWD=passwd;ConnectionName=One
```

For more information on .NET connection pooling, refer to the topic on .NET connection pooling.

Example

The following connection string fragment turns connection pooling off.

```
ConnectionPool=NO;
```

The following connection string fragment turns connection pooling on with a maximum of 10 cached connections.

```
CPOOL=YES(MaxCached=10);
```

Related Information

Improve Application Performance with Connection Pooling [page 256]

Connection Parameters and Connection Strings [page 35]
.NET Connection Pooling

1.1.3.13 DatabaseFile (DBF) Connection Parameter

Indicates which database file you want to load and connect to when starting a database that is not running.

Syntax

```
{ DatabaseFile | DBF }=filename
```

Usage

Local database servers

Allowed Values

filename

This string specifies the path and file name of a database to start.

If the file name does not include an extension, the database server looks for a file with the `.db` extension.

The path of the file is relative to the working directory of the database server. If you start the database server at a command prompt, the working directory is the directory that you are in when you run the command. If you start the database server from an icon or shortcut, it is the working directory that the icon or shortcut specifies. It is recommended that you supply a complete path and file name.

UNC file names are supported.

Default

There is no default setting.

Remarks

If you are unsure whether the database is running, you can use the DatabaseFile (DBF) connection parameter to start the database and connect to it.

If you specify both the DatabaseFile (DBF) and DatabaseName (DBN) connection parameters, an attempt is made to connect to a running database with the specified database name (the DatabaseFile connection parameter is ignored). If that fails, an attempt is made to start the database automatically using the path indicated by the DatabaseFile connection parameter and assign it the name specified by the DatabaseName connection parameter.

If you specify the DatabaseFile (DBF) connection parameter, but not the DatabaseName (DBN) connection parameter, an attempt is made to connect to a running database with the same name as the file specified (the path and extension are removed). If that attempt fails, an attempt is made to start the database automatically using the path indicated by the DatabaseFile connection parameter.

A database cannot be automatically started on a network server unless this behavior is explicitly permitted by the -gd database server option.

If a running database server cannot be found, by default, a personal database server is automatically started.

For deployed applications, it is recommended that you specify a database server name using the ServerName (Server) parameter when attempting to start a database file automatically if it is not running. Otherwise, the application may connect to a different database server than intended. For example, the database server could connect to a different version of the SQL Anywhere database server that is part of an embedded application and already running.

> ⚠ Caution
>
> The database file must be on the same computer as the database server. Starting a database file that is located on a network drive can lead to file corruption.

Example

The DatabaseFile (DBF) connection parameter in the following example loads and connects to the sample database, *demo.db*:

```
"DBF=%SQLANYSAMP17%\demo.db"
```

The following two examples assume that you have started a database file named `cities.db`, and renamed the database Kitchener as follows:

```
dbeng17 cities.db -n Kitchener
```

To successfully start and connect to a database and name it Kitchener specify the following DBN and DBF connection parameter values:

```
DBN=Kitchener;DBF=cities.db
```

Specifying `DBF=cities.db` fails to connect to the running database named Kitchener.

Related Information

Connection Parameters and Connection Strings [page 35]

Connection Parameters to Use with Embedded Databases [page 42]
-gd Database Server Option [page 430]
CommLinks (LINKS) Connection Parameter [page 59]
DatabaseName (DBN) Connection Parameter [page 71]

1.1.3.14 DatabaseKey (DBKEY) Connection Parameter

Starts an encrypted database with a connect request.

Syntax

```
{ DatabaseKey | DBKEY }=key
```

Usage

Anywhere

Allowed Values

key

The encryption key is a string, including mixed cases, numbers, letters, and special characters. Database keys cannot include leading spaces, trailing spaces, or semicolons.

Default

None

Remarks

You must specify this parameter when you start an encrypted database with a connect request.

To secure communication packets between client applications and the database server, use the -ec server option and transport layer security.

Example

The following fragment illustrates the use of the DatabaseKey (DBKEY) connection parameter:

```
"UID=DBA;PWD=sql;Server=myeng;DBKEY=V3moj3952B;DBF=%SQLANYSAMP17%\demo.db"
```

Related Information

Transport Layer Security [page 1727]
Client Application Configuration to Use Transport Layer Security [page 1747]
Connection Parameters and Connection Strings [page 35]
-ec Database Server Option [page 418]
-ek Database Option [page 551]
-ep Database Server Option [page 423]
-es Database Server Option [page 424]
Encryption (ENC) Connection Parameter [page 80]

1.1.3.15 DatabaseName (DBN) Connection Parameter

Identifies a running database or assigns a name to a database that is starting.

Syntax

```
{ DatabaseName | DBN }=database-name
```

Usage

Anywhere

Allowed Values

database-name

This string specifies the name of a database that is already running on a database server or names a database that is starting.

Default

There is no default setting.

Remarks

Whenever a database is started on a server, it is assigned a database name, either by the administrator by using the -n option, or by the database server by using the base of the file name with the extension and path removed.

Connect to a running database

If the database you want to connect to is already running, use the DatabaseName (DBN) connection parameter. A connection only occurs if the name of the running database matches `database-name`.

To connect to the utility database, use the database name utility_db.

The DatabaseName (DBN) connection parameter is recommended over the DatabaseFile (DBF) connection parameter when you want to prevent any attempt to start the database if it is not already running.

Start a database that is not running

If you specify both the DatabaseName (DBN) and DatabaseFile (DBF) connection parameters, an attempt is made to connect to a running database with the specified database name (the DatabaseFile connection parameter is ignored), and if that fails, an attempt is made to start the database automatically using the path indicated by the DatabaseFile connection parameter and assign it the name specified by the DatabaseName connection parameter.

The DatabaseName (DBN) connection parameter is recommended for naming databases, rather than using the DatabaseSwitches (DBS) connection parameter with the -n option or the START connection parameter with the -n option.

Example

To start a database named `cities.db` and name the database Kitchener, use the following connection string:

```
"DBF=cities.db;DBN=Kitchener;UID=DBA;PWD=passwd"
```

To connect to a running database named Kitchener, use the following connection string:

```
"DBN=Kitchener;UID=DBA;PWD=passwd"
```

Related Information

Connection Parameters and Connection Strings [page 35]

The Utility Database (utility_db) [page 309]
Common Connection Scenarios [page 36]
DatabaseName (DBN) Protocol Option [page 201]
DatabaseSwitches (DBS) Connection Parameter [page 73]

1.1.3.16 DatabaseSwitches (DBS) Connection Parameter

Provides database-specific options when starting a database.

Syntax

```
{ DatabaseSwitches | DBS }=database-options
```

Usage

Anywhere

Allowed Values

database-options

This string specifies database options that apply to the database file.

To supply the database server options use the StartLine connection parameter.

Default

No options.

Remarks

Use this connection parameter to connect to a database server and start a database on it. If the database server is not running, this connection parameter starts the database server automatically with the specified database and options.

If the database is running before the connection attempt, the DatabaseSwitches connection parameter is ignored.

When the database server starts the database specified by DatabaseFile, the server uses the supplied DatabaseSwitches to determine startup options for the database.

> **i** Note
>
> The DatabaseName (DBN) connection parameter is recommended for naming databases, rather than using the DatabaseSwitches (DBS) connection parameter with the -n option.

Example

The following command, connects to the default database server, loads the database file *demo.db* (DatabaseFile (DBF) connection parameter), names it my-db (DatabaseName (DBN) connection parameter) and starts it in read-only mode (-r option).

```
dbisql -c "UID=DBA;PWD=sql;DBF=%SQLANYSAMP17%\demo.db;DBN=my-db;DBS=-r"
```

Related Information

Connection Parameters and Connection Strings [page 35]
SQL Anywhere Database Server Executable (dbsrv17, dbeng17) [page 380]
Database Startup Options [page 543]
DatabaseName (DBN) Connection Parameter [page 71]
DatabaseKey (DBKEY) Connection Parameter [page 70]
StartLine (START) Connection Parameter [page 112]

1.1.3.17 DataSourceName (DSN) Connection Parameter

Tells the ODBC driver manager or Embedded SQL library where to look in the registry or the system information file (named `.odbc.ini` by default) to find ODBC data source information.

> **⁝≡** Syntax
>
> { *DataSourceName* | *DSN* }=data-source-name

Usage

Anywhere

Allowed Values

data-source-name

This string specifies the name of the ODBC data source that contains connection in formation for your database.

Default

There is no default data source name.

Remarks

It is common practice for ODBC applications to send only a data source name to ODBC. The ODBC driver manager and ODBC driver locate the data source, which contains the remainder of the connection parameters.

Embedded SQL applications can also use ODBC data sources to store connection parameters.

Example

The following parameter uses a data source name:

```
DSN=My Database
```

Related Information

Connection Parameters and Connection Strings [page 35]
ODBC Data Sources on UNIX/Linux [page 126]
ODBC Data Sources [page 116]
FileDataSourceName (FILEDSN) Connection Parameter [page 84]

1.1.3.18 DisableMultiRowFetch (DMRF) Connection Parameter

Disables multi-row prefetch.

Syntax

```
{ DisableMultiRowFetch | DMRF }={ YES | NO }
```

Usage

Anywhere

Default

NO

Remarks

By default, when an application fetches a single row from the database server, the client library anticipates that additional rows will be fetched and asks for extra rows. This is called prefetch and it is used to improve performance. You can disable prefetch by setting this parameter to YES.

Setting the DisableMultiRowFetch (DMRF) connection parameter to YES is equivalent to setting the prefetch database option to Off.

This option does not disable wide (array, multi-row) fetches. It only affects prefetch.

Example

The following connection string fragment prevents prefetching:

```
DMRF=YES
```

Related Information

Connection Parameters and Connection Strings [page 35]

Cursors in Procedures, Triggers, User-defined Functions, and Batches
Prefetches
prefetch Option [page 794]

1.1.3.19 Elevate Connection Parameter

Elevates automatically started database server executables on Microsoft Windows.

Syntax

```
Elevate={ YES | NO }
```

Usage

Microsoft Windows

Default

NO

Remarks

Specifying ELEVATE=YES in your connection string setting allows non-elevated client processes to start elevated servers automatically, which may be necessary on Windows 7 or later if administrative rights are required to access the database file. This parameter is ignored if the database server is not started automatically.

Example

The following connection string fragment starts a database server automatically and elevates:

```
"Elevate=YES;START=dbsrv17"
```

1.1.3.20 EncodedPassword (ENP) Connection Parameter

Indicates that the password string is in an encoded form.

Syntax

```
{ EncodedPassword | ENP }=password
```

Usage

Anywhere

Allowed Values

password

The value is the hexadecimal representation of a binary encoded password.

Default

None

Remarks

When you connect to a database using Standard user authentication, provide a user ID and password to the database server. Other supported user authentication frameworks such as Kerberos, LDAP (Lightweight Directory Access Protocol), PAM (Pluggable Authentication Module), and Microsoft Windows integrated login do not require a password for the database connection.

Although best practice suggests that the password be supplied by the user on demand (for example, by prompting), there may be situations where you must store the user's credentials on the client computer. Storing passwords on the client computer poses a security risk. If both the user ID and the password are discovered, then anyone with access to the database server can authenticate with that user ID and password. For these situations, the client software provides a mechanism for storing passwords in plain text or in an encoded form.

In this topic, the term *encode* refers to the process of transforming a plain text string, either through obfuscation or through encryption, into an encoded form. The term *obfuscate* refers to the process of transforming a plain text string in a manner that is not cryptographically secure into an obfuscated form. The term *encrypt* refers to the process of transforming a plain text string in a manner that is cryptographically secure into an encrypted form.

The EncodedPassword connection parameter is used in a connection string to indicate that the password is encoded. The encoded string is in hexadecimal format.

To create an encoded password, use the *ODBC Configuration For SQL Anywhere* window in the *ODBC Data Source Administrator* (Microsoft Windows only) or the Data Source utility (dbdsn).

Using one of these tools, the encoded password is stored in the ENP (EncodedPassword) parameter field in a data source as a hexadecimal string.

On Microsoft Windows, data source information is usually stored in the system registry. On UNIX and Linux, data source information is stored in a system information file (named `.odbc.ini` by default).

When an encoded password is provided as part of a connect request, it is decoded back to plain text by the client software before any connection attempt is made.

If both the Password (PWD) connection parameter and the EncodedPassword (ENP) connection parameter are specified, then the Password (PWD) connection parameter takes precedence.

Different levels of encoded password are supported.

1. The password is obfuscated for use on any computer. The encoded password string and its corresponding user ID can be used by anyone on any computer to initiate a connection to a database.
2. The password is encoded on a computer such that it can only be decoded on that computer. The encoded password string and its corresponding user ID can only be used to authenticate to a database from the computer where the encoded password was created. Anyone who can log on to the computer can use the encoded password and its corresponding user ID to authenticate to a database. It cannot be used on any other computer.
3. The password is encoded on a computer by a user such that it can only be decoded on that computer by that user only. The encoded password string and its corresponding user ID can only be used to authenticate to a database from the computer where the encoded password was created by the user who created it. It cannot be decoded by other users of that computer. It cannot be decoded on any other computer by the same or other user.

Specify the level of encoding to use for the password that is stored in an ODBC data source with the *ODBC Configuration For SQL Anywhere* window in the *ODBC Data Source Administrator* (Microsoft Windows only) or with the -pet option of the Data Source utility (dbdsn).

⚠ Caution

When you create a data source, do not include the user ID and either the plain text password or the encoded password as part of the definition. Although both the *ODBC Configuration For SQL Anywhere* window in the Microsoft Windows *ODBC Data Source Administrator* and the Data Source utility (dbdsn) have this capability, including this information poses a security risk. If you must include the password, then choose the best form of encoding possible.

On Microsoft Windows, the Microsoft Cryptography API is used to provide strong encryption for levels 2 and 3 described above. Microsoft Windows manages the encryption key that is used to encrypt the password using the integrated key store. The encryption key is linked to a specific computer or a specific user and computer. Any user on the computer can still decode a level-2 encoded password into plain text. The specific user on the computer can still decode a level-3 encoded password into plain text. No password (other than the one the user provided when logging into Microsoft Windows) is required for this decoding to be possible.

Although level 3 can be used for a Microsoft Windows System ODBC data source, it may be more appropriate to use level 2 since this form of encrypted password can be used by any user of the computer.

If the system ODBC data source is created and used by only one Microsoft Windows user, then level 3 may be used for a system ODBC data source.

On Linux, macOS, and other systems, there is no integrated key store. As such, only obfuscation is provided at levels 2 and 3. Obfuscation cannot be considered secure in the face of attackers trying to obtain the password. You achieve operating system level protection by setting permissions on the `.odbc.ini` file appropriately.

The encoded password binary value is always specified in its hexadecimal representation. The following is an example.

```
ENP=03a17731bca92f9701000000d08c9ddf0115d1118c7a00c04fc297eb01...
```

An advantage to a password encoded for a specific user and/or computer is that, even if the password is accidentally disclosed, it cannot be used on another computer system using the client library. Note, however, that an attacker may be able to decode it depending on the level chosen and the operating system platform.

Related Information

ODBC Data Sources on UNIX/Linux [page 126]
Connection Parameters and Connection Strings [page 35]
Data Source Utility (dbdsn) [page 1140]
Password (PWD) Connection Parameter [page 104]

1.1.3.21 Encryption (ENC) Connection Parameter

Specifies the encoding for packets sent between the client application and the database server. This may be no encoding, simple obfuscation, or TLS (Transport Layer Security) encryption.

Syntax

```
{ Encryption | ENC }=
 { NONE
 | SIMPLE
 | TLS [ (
    [ FIPS={ ON | OFF }; ]
    [ TRUSTED_CERTIFICATE={ public-certificate-filename | * | NONE } ; ]
    [ CERTIFICATE_COMPANY=organization; ]
    [ CERTIFICATE_NAME=common-name; ]
    [ CERTIFICATE_UNIT=organization-unit; ]
    [ SKIP_CERTIFICATE_NAME_CHECK={ ON | OFF } ]
    [ IDENTITY=identity-file ]
    [ IDENTITY_PASSWORD=password ]
    [ TLS_TYPE=rsa ]
    [ DIRECT={ YES | NO } ]
    [ MIN_TLS_VERSION=ver ]
 ) ]
 }
```

Usage

TLS: supported for TCP/IP only

NONE or SIMPLE: anywhere

Allowed Values

NONE

Accepts communication packets that are not encrypted.

SIMPLE

Accepts communication packets that are encoded with simple obfuscation. This value is supported on all platforms and on previous versions of SQL Anywhere. Simple obfuscation does not provide database server verification, strong encryption, or other features of Transport Layer Security.

If the database server accepts simple obfuscation, but does not accept unencrypted communication, then any non-TDS connection attempts using no encryption automatically use simple obfuscation.

Starting the database server with `-ec SIMPLE` tells the database server to accept only connections using simple obfuscation. TLS connections requesting RSA encryption or FIPS-certified RSA encryption fail, and connections requesting no encryption use simple obfuscation.

TLS

Accepts communication packets that are encrypted using TLS. For FIPS-certified TLS, specify FIPS=ON. FIPS-certified TLS uses a separate certified library, but is compatible with SQL Anywhere 9.0.2 or later database servers using non-certified TLS.

When initiating TLS connections, the client libraries will check the host name of the database server against the certificate provided by that server. This check will only happen if none of the certificate_name, certificate_company, or certificate_unit options are specified, or the skip_certificate_name_check option is OFF. If any of certificate_name, certificate_company, or certificate_unit are specified, only those options are verified. The skip_certificate_name_check option disables the host name check when ON.

The host names or IP addresses are derived from the subjectAltName (Subject Alternative Name or SAN) extension and the Common Name (CN) field. The SAN may contain multiple host names with wild cards. For example, a Google certificate might include *.google.com, *.google.ca, and *.android.com. Thus, www.google.ca is a valid host name.

The client supports the following arguments to verify the field values in the database server's certificate, whether you are using a public certificate or a certificate from the operating system's certificate store.

- CERTIFICATE_COMPANY
- CERTIFICATE_NAME
- CERTIFICATE_UNIT

TLS and TLS(TRUSTED_CERTIFICATE=*) are equivalent, and specifying either causes the software to use the operating system certificate store and trust every certificate listed in the store.

> **i Note**
>
> Specify *NONE* for the trusted_certificate protocol option to make a TLS connection without verifying the server's certificate. Connecting without verifying the server's certificate is less secure than

verifying the certificate because the client can no longer protect against a man-in-the-middle attack. However, the connection is still strongly encrypted and prevents replay attacks, which is more secure than no encryption at all. With the *none* option, no trusted root certificate is required on the client.

Specify SKIP_CERTIFICATE_NAME_CHECK=ON to prevent the host name check. Setting this option to ON may prevent the client from fully authenticating the database server.

Use the IDENTITY and IDENTITY_PASSWORD option to specify an identity file and password for the client (not the database server).

TLS_TYPE specifies the encyption type to use for the connection. The only supported value is *rsa*.

Specify DIRECT=YES when connecting through a proxy. You cannot use DIRECT=YES with the SERVER or ENG parameters, or with the LINKS parameter if you specify VERIFY=YES.

Use MIN_TLS_VERSION to specify the minimum downgrade TLS version. The supported values for `ver` are *1.0*, *1.1*, and *1.2* (the default). The period is optional, so you can also specify *10*, *11*, or *12*.

Default

NONE

Remarks

This parameter can be used to encode communications between client applications and the database server using Transport Layer Security (TLS) or simple obfuscation.

FIPS-certified encryption requires a separate license.

For one-way authentication in which the client authenticates the database server, use the TRUSTED_CERTIFICATE option. If you want to enable two-way authentication in which the database server also authenticates the client, then use the IDENTITY and IDENTITY_PASSWORD option to specify an identity file and password for the client. The database server specifies the trusted certificate that was used to sign the client's identity file.

Use the CONNECTION_PROPERTY system function to retrieve the encryption settings for the current connection:

```
SELECT CONNECTION_PROPERTY ( 'Encryption' );
```

Example

The following connection string fragment connects to a database server using Transport Layer Security (RSA encryption) and a specified certificate file that is accepted even if it is expired:

```
"HOST=myhost;SERVER=myserver;ENCRYPTION=TLS(FIPS=OFF;TRUSTED_CERTIFICATE=rsaroot.
crt)"
```

The following connection string fragment connects to a database server using simple obfuscation:

```
"HOST=myhost;SERVER=myserver;ENCRYPTION=simple"
```

The following two connection string fragments are equivalent and connect to a database server using Transport Layer Security (RSA encryption) and a certificate from the operating system certificate store:

```
"HOST=myhost;SERVER=myserver;ENCRYPTION=TLS(FIPS=OFF;TRUSTED_CERTIFICATE=*)"
```

```
"HOST=myhost;SERVER=myserver;ENCRYPTION=TLS"
```

The following connection string fragment verifies that `myrootcert.crt` is at the root of the database server's certificate's signing chain and that the common name field of the certificate is set to MyCertificateName:

```
"HOST=myhost;SERVER=myserver;ENCRYPTION=TLS(TRUSTED_CERTIFICATE=myrootcert.crt;CE
RTIFICATE_NAME=MyCertificateName)"
```

The following connection string fragment verifies that `myrootcert.crt` is at the root of the database server's certificate's signing chain. Since no other TLS certificate option is specified, the host name in the database server's certificate must match the host name being connected to:

```
"HOST=myhost;SERVER=myserver;ENCRYPTION=TLS(TRUSTED_CERTIFICATE=myrootcert.crt)"
```

The following connection string fragment verifies that `myrootcert.crt` is at the root of the database server's certificate's signing chain, but does no other verification:

```
"HOST=myhost;SERVER=myserver;ENCRYPTION=TLS(TRUSTED_CERTIFICATE=myrootcert.crt;SK
IP_CERTIFICATE_NAME_CHECK=ON)"
```

The following connection string fragment forces the minimum TLS version to 1.2 to avoid weak TLS 1.0 and 1.1:

```
"HOST=myhost;SERVER=myserver;ENCRYPTION=TLS(MIN_TLS_VERSION=1.2)"
```

Related Information

Client Application Configuration to Use Transport Layer Security [page 1747]
Connection Parameters and Connection Strings [page 35]
Transport Layer Security [page 1727]
Digital Certificates [page 1737]
Server Authentication
-ec Database Server Option [page 418]
-ek Database Option [page 551]
-ep Database Server Option [page 423]
-es Database Server Option [page 424]
DatabaseKey (DBKEY) Connection Parameter [page 70]
certificate_company Protocol Option (Client Side Only) [page 195]
certificate_name Protocol Option (Client Side Only) [page 196]
certificate_unit Protocol Option (Client Side Only) [page 198]

trusted_certificate Protocol Option [page 244]
CONNECTION_PROPERTY Function [System]
skip_certificate_name_check Protocol Option (Client Side Only) [page 238]

1.1.3.22 EngineName (ENG) Connection Parameter (Deprecated)

This connection parameter is deprecated. Use the ServerName (Server) connection parameter instead.

Related Information

ServerName (Server) Connection Parameter [page 110]

1.1.3.23 FileDataSourceName (FILEDSN) Connection Parameter

Tells the client library there is an ODBC file data source holding information about the database to which you want to connect.

Syntax

```
{ FileDataSourceName | FILEDSN }=file-data-source-name
```

Usage

Anywhere

Allowed Values

file-data-source-name

This string specifies the name of the File Data Source that contains connection in formation for your database.

Default

There is no default name.

Remarks

File Data Sources hold the same information as ODBC data sources stored in the registry. File Data Sources can be easily distributed to end users so that connection information does not have to be reconstructed on each computer.

Both ODBC and Embedded SQL applications can use file data sources.

Related Information

File Data Sources on Microsoft Windows [page 124]
Connection Parameters and Connection Strings [page 35]
File Data Sources on Microsoft Windows [page 124]
DataSourceName (DSN) Connection Parameter [page 74]

1.1.3.24 ForceStart (FORCE) Connection Parameter

Starts a database server without checking if any server is running.

Syntax

```
{ ForceStart | FORCE }={ YES | NO }
```

Usage

Only with the db_start_engine function.

Default

NO

Remarks

By setting ForceStart=YES, the db_start_engine function does not check if any server is running before starting the server.

Related Information

Connection Parameters and Connection Strings [page 35]
db_start_engine Function

1.1.3.25 Host Connection Parameter

Accepts a host name or IP address and optional port number that tells the client where to find the database server.

Syntax

```
Host={ hostname | ip-address }[ :port-number ] [, ...]
```

Usage

Anywhere. The Host connection parameter is recommended for connections to a network database server and indicates the use of TCP/IP.

Allowed Values

hostname

The name of the computer where the database server is running. The list of host values is a comma-separated list and it can optionally include a port number (separated by a colon). You can use **localhost** to identify the current computer.

ip-address

This string must be specified in the form of an IP address and it can optionally include a port number (separated by a colon). The list of IP addresses is a comma-separated list.

For IPv6 addresses that include a port number, you must enclose the address in either square brackets or parentheses. For example, `Host=[fd77:ab34:2238::3894]:8933`, where 8933 is the port number.

port-number

The port number used by the database server. The default port number is 2638.

Default

None

Remarks

The Host connection parameter specifies one or more host names (or IP addresses) and optional port numbers that tell the client where a database server is running.

The Host connection parameter identifies one or more computer systems running database servers. A database server is not started automatically when the Host connection parameter is specified.

If a computer system is running more than one database server and a port is not specified, the ServerName connection parameter should be used to identify which database server to connect to. It is recommended that you always use the ServerName parameter, particularly for embedded applications.

Option	Description
Specify server and port	`host=myhost:1234;servername=myserver` A connection is made only to a database server listening on the specified port and with the specified server name.
Specify server, but not port	`host=myhost;servername=myserver` A connection is made to the database server with the specified server name.
Specify neither server nor port	`host=myhost` A connection is attempted to a database server listening on port 2638.
Specify multiple servers and ports	`host=myhost1:6871,myhost2:6872` A connection is made to a database server listening on one of the specified hosts/ports.

When multiple addresses are specified, the addresses are attempted in the following order:

1. The specified addresses are checked against the database server address cache (`sasrv.ini`). If a match is found, then this address in the database server cache is attempted first.
2. The addresses are attempted in the order in which they are specified.

When you use the Host parameter, no UDP packets are sent if enough information is given to uniquely identify the server (a host name and a port number). If neither a port number nor database server name is given, the port number is assumed to be 2638 and the client does not perform a broadcast. However, if the client has a host name and database server name but no port number, it sends a UDP packet to port 2638 on the specified host to find the port number.

> **i Note**
>
> The Host connection parameter must be specified when using TLS encryption and the host name must match the Common Name in the database server's identity certificate, unless the skip_certificate_name_check option is enabled in the Encryption (ENC) connection parameter. The CommLinks (LINKS) connection parameter cannot be used as an alternative in this case.
>
> The Host connection parameter disables the use of shared memory. To use the Host connection parameter to connect to a personal database server, you must enable the TCP/IP communication protocol on the database server using the -x option. Also, you must use the host name **localhost** when connecting to a personal database server using TCP/IP.
>
> It is recommended that you only use the CommLinks (LINKS) connection parameter if you need to specify TCP/IP protocol options other than HOST or ServerPort (PORT).
>
> You cannot specify both CommLinks and Host in a connection string.

The CommLinks connection parameter requires the specification of a host name, but the port number and the ServerName connection parameter are optional. Therefore, there are four possible combinations:

Host connection parameter	Equivalent CommLinks (LINKS) connection parameter string
`Host=serverhost:1234` A connection attempt is made to a database server running on serverhost port 1234.	`LINKS=TCPIP(HOST=serverhost:1234;DoBroadcast=None;VerifyServerName=No)`
`Host=serverhost:1234;ServerName=myserver` A connection attempt is made to a database server named myserver running on serverhost port 1234.	`LINKS=TCPIP(HOST=serverhost:1234;DoBroadcast=None;VerifyServerName=Yes);ServerName=myserver`
`Host=serverhost` A connection attempt is made to a database server running on serverhost on the default port, 2638.	`LINKS=TCPIP(HOST=serverhost:2638;DoBroadcast=None;VerifyServerName=No)`
`Host=serverhost;ServerName=myserver` A connection attempt is made to a database server named myserver running on serverhost on any port.	`LINKS=TCPIP(HOST=serverhost;DoBroadcast=Direct;VerifyServerName=Yes);ServerName=myserver`

Example

If you know that a database server named SalesDB is running on a computer called Elora on the default port number, then you can use the following connection string to connect to the database server:

```
UID=DBA;PWD=passwd;Server=SalesDB;Host=Elora:2638
```

If you do not know the port number that the database server is running on, use the following connection string to connect to the SalesDB database server running on the computer named Elora:

```
UID=DBA;PWD=passwd;Server=SalesDB;Host=Elora
```

Related Information

Common Connection Scenarios [page 36]
Host (IP) Protocol Option (Client Side Only) [page 206]
CommLinks (LINKS) Connection Parameter [page 59]
ServerName (Server) Connection Parameter [page 110]
Troubleshooting: How the Host Connection Parameter Locates Database Servers [page 267]

1.1.3.26 IdleTimeout (IDLE) Connection Parameter

Specifies a connection's idle timeout period.

Syntax

```
{ IdleTimeout | IDLE }=timeout-value
```

Usage

Anywhere except with TDS connections. TDS connections (including jConnect) ignore the IdleTimeout connection parameter.

Allowed Values

timeout-value

The connection's idle timeout period, in minutes. The minimum value for the IdleTimeout connection parameter is 1 minute, and the maximum supported value is 32767 minutes. If you specify 0, idle timeout checking is turned off for the connection.

Default

240 minutes (TCP/IP)

0 (shared memory)

Remarks

The IdleTimeout connection parameter applies to the current connection. You can have multiple connections to the same database server with different timeout values.

If the IdleTimeout connection parameter is not used, then the idle timeout value for TCP/IP connections is controlled by the -ti database server option. If both the -ti database server option and the IdleTimeout connection parameter are specified, then the idle timeout value is controlled by the connection parameter.

IdleTimeout also applies to pooled connections. A pooled connection will be terminated (closed) when the timeout period elapses. The lifetime of a connection is calculated by taking the difference between the current time and the last time the connection was busy.

Example

The following connection string fragment sets the timeout value for this connection to 10 minutes:

```
"Host=myhost;IDLE=10"
```

Related Information

Connection Parameters and Connection Strings [page 35]
Automatically Release Schema Locks (Interactive SQL) [page 1085]
-ti Database Server Option [page 505]
Troubleshooting Network Communications [page 1028]
blocking_others_timeout Option [page 695]

1.1.3.27 Integrated (INT) Connection Parameter

Specifies whether an integrated login can be attempted.

Syntax

```
{ Integrated | INT }={ YES | NO }
```

Usage

Integrated logins are only permitted when both the client and server machines are running on Windows systems.

Allowed Values

YES

An integrated login is attempted. If the connection attempt fails and the login_mode option is set to 'Standard,Integrated', a standard login is attempted.

NO

This is the default setting. No integrated login is attempted.

Default

NO

Remarks

For a client application to use an integrated login, the server must be running with the login_mode database option set to a value that includes Integrated.

Example

The following data source fragment uses an integrated login:

```
INT=YES
```

Related Information

Connection Parameters and Connection Strings [page 35]
Microsoft Windows Integrated Login [page 129]
login_mode Option [page 752]

1.1.3.28 Kerberos (KRB) Connection Parameter

Specifies whether Kerberos authentication can be used when connecting to the database server.

Syntax

```
{ Kerberos | KRB }= { YES | NO | SSPI | GSS-API-library-file }
```

Usage

All platforms.

Allowed Values

YES

A Kerberos authenticated login is attempted.

NO

No Kerberos authenticated login is attempted. This is the default.

SSPI

A Kerberos authenticated login is attempted, and the built-in Windows SSPI interface is used instead of a GSS-API library. SSPI can only be used on Windows platforms, and it cannot be used with a Key Distribution Center (KDC) other than the Domain Controller Active Directory KDC. If your Windows client computer has already logged in to a Windows domain, SSPI can be used without needing to install or configure a Kerberos client.

> **Note**
>
> SSPI can only be used by clients in the Kerberos connection parameter. The database server cannot use SSPI. It needs a supported Kerberos client other than SSPI.

GSS-API-library-file

A Kerberos authenticated login is attempted, and this string specifies the file name of the Kerberos GSS-API library (or shared object on UNIX and Linux). This is only required if the Kerberos client uses a different file name for the Kerberos GSS-API library than the default, or if there are multiple GSS-API libraries installed on the computer.

Default

NO

Remarks

The UserID and Password connection parameters are ignored when using a Kerberos authenticated login.

To use Kerberos authentication, a Kerberos client must already be installed and configured (nothing needs to be done for SSPI), the user must have already logged in to Kerberos (have a valid ticket-granting ticket), and the database server must have enabled and configured Kerberos authenticated logins.

Example

```
Kerberos=YES
Kerberos=SSPI
Kerberos=c:\Program Files\MIT\Kerberos\bin\gssapi32.dll
```

Related Information

Kerberos User Authentication [page 159]
Connecting Using SSPI for Kerberos Logins on Microsoft Windows [page 168]
-kl Database Server Option [page 457]
-kr Database Server Option (Deprecated) [page 459]
-krb Database Server Option [page 461]

1.1.3.29 Language (LANG) Connection Parameter

Specifies the language of the connection.

Syntax

```
{ Language | LANG }=language-code
```

Usage

Anywhere

Allowed Values

language-code

The two-letter combination representing a language. For example, specifying LANG=DE sets the default language to German.

Default

The language specified by (in order) the SALANG environment variable, the dblang utility, or the installer.

Remarks

This connection parameter establishes the language for the connection. Any errors or warnings from the server are delivered in the specified language, assuming that the server supports the language.

If no language is specified, the default language is used. The default language is the language specified by, in order, the SALANG environment variable, the dblang utility, or the installer.

This connection parameter only affects the connection. Messages returned from tools and utilities appear in the default language, while the messages returned from the server appear in the connection's language.

Related Information

Connection Parameters and Connection Strings [page 35]
Language Label Values and Language Codes [page 615]

1.1.3.30 LazyClose (LCLOSE) Connection Parameter

Controls whether cursor requests are queued until the next request or performed immediately.

Queuing close cursor requests saves a round trip and improves performance.

Syntax

```
{ LazyClose | LCLOSE }={ YES | NO | AUTO }
```

Usage

Anywhere

Allowed Values

YES

Always queue the cursor close request, which saves a round trip, but can cause locks and other resources to be held after the cursor is closed by the client. The cursor close is performed when the next request is sent to the database server on the same connection. Any isolation level 1 cursor stability locks still apply to the cursor while the `CLOSE cursor-name` database request is queued.

NO

Close the cursor immediately.

AUTO

Queue the cursor close request and save a round trip, only when doing so doesn't change how long locks or significant server resources are held. If the cursor uses isolation level 1 cursor stability locks, or could consume significant server resources that are not released until the cursor is closed, then the cursor is closed immediately. A query that requires a work table is an example of a cursor that can consume significant server resources.

Default

AUTO

Remarks

When this connection parameter is set to YES or AUTO, cursors are not closed until the next database request.

Enabling this option can improve performance, if your network exhibits poor latency or your application sends many cursor open and close requests.

Related Information

Connection Parameters and Connection Strings [page 35]

1.1.3.31 LivenessTimeout (LTO) Connection Parameter

Controls the shutdown of connections when they are no longer intact.

Syntax

```
{ LivenessTimeout | LTO }=timeout-value
```

Usage

Network server only.

All platforms except non-threaded Unix applications.

Allowed Values

timeout-value

The connection's liveness timeout period, in seconds. The minimum value for the LivenessTimeout connection parameter is 30 seconds, and the maximum value is 32767 seconds. If you specify 0, liveness timeout checking is turned off for the connection. Any non-zero value less than the minimum value is reset to the minimum value. For example, a connection string containing "LivenessTimeout=5" uses "LivenessTimeout=30". If no LivenessTimeout value is set, the LivenessTimeout is controlled by the setting on the server, which defaults to 120 seconds.

Default

None

Remarks

A **liveness packet** is sent periodically across a client/server TCP/IP communication protocol to confirm that a connection is intact. If the client runs for the LivenessTimeout period without detecting a liveness request or response packet, the communication is ended.

Liveness packets are sent when a connection has not sent any packets for between one third and two thirds of the LivenessTimeout value.

When there are more than 200 connections to a server, the server automatically calculates a higher LivenessTimeout value based on the stated LivenessTimeout value. This enables the server to handle a large number of connections more efficiently.

Alternatively, you can set this parameter by entering its value in the *LivenessTimeout* text box on the *Advanced* tab of the *ODBC Configuration For SQL Anywhere* window.

Example

The following connection string fragment sets a LivenessTimeout value of 10 minutes:

```
LTO=600
```

Related Information

Connection Parameters and Connection Strings [page 35]
-tl Database Server Option [page 506]

1.1.3.32 LogFile (LOG) Connection Parameter

Sends client error messages and debugging messages to a file.

Syntax

```
{ LogFile | LOG }=filename
```

Usage

Anywhere

Allowed Values

filename

This string specifies the name of the file where client error messages and debugging messages are saved. If the file name does not include a path, it is relative to the current working directory of the client application.

Default

No log file.

Remarks

The LogFile (LOG) connection parameter is connection-specific, so from a single application you can set different LogFile arguments for different connections.

This connection parameter is only supported for CmdSeq connections.

Typical log file contents are as follows:

```
Tue Jan 22 2013 10:59:3
10:59:31 Attempting to connect using:
UID=DBA;PWD=********;DBF='C:\Users\Public\Documents\SQL Anywhere
         17\Samples\demo.db';ServerName=demo17;START='C:\Program Files\SQL
Anywhere 17\Bin32\dbeng17.exe';CON=SQL_DBC_10b9d400;ASTOP=YES;LOG=c:\logs
\test.txt
10:59:31 Attempting to connect to a running server...
10:59:31 Attempting SharedMemory connection (no sasrv.ini cached address)
10:59:31 Connected to server over SharedMemory
10:59:31 Connected to SQL Anywhere Server version 17.0.11.1293
10:59:31 Application information:
10:59:31 IP=192.138.151.100;HOST=MyComputer1-PC;OSUSER=user1;OS='Windows 7 Build
7601 Service Pack 1';EXE='C:\Program Files\SQL Anywhere
17\bin32\dbisql.exe';PID=0x145c;THREAD=0x508;VERSION=17.0.11.1293;API=iAnywhereJD
BC;TIMEZONEADJUSTMENT=-300
10:59:31 Connected to the server, attempting to connect to a running database...
10:59:31 [  205] Connected to database successfully
10:59:31 [  205] The number of prefetch rows has been reduced to 16 due to the
prefetch buffer
10:59:31 [  205] limit. Increasing the PrefetchBuffer connection parameter may
improve performance.
```

Example

The following command line starts Interactive SQL, and connects to the sample database with a LogFile (LOG) connection parameter:

```
dbisql -c "DSN=SQL Anywhere 17 Demo;PWD=sql;LOG=c:\logs\test.txt"
```

Related Information

Connection Parameters and Connection Strings [page 35]
LogFile (LOG) Protocol Option [page 217]

1.1.3.33 NewPassword (NEWPWD) Connection Parameter

Allows users to change passwords, even if they have expired, without DBA intervention.

Syntax

```
{ NewPassword | NEWPWD }={ password-string | * }
```

Usage

The client library prompting for a new password is only supported on Microsoft Windows.

Allowed Values

password-string

If the user provides a new password, the database server authenticates the user ID and password and attempts to change the password before the login_procedure option is called. This process allows the user to change an expired password without the involvement of a DBA. If you have set the verify_password_function option, the new password is verified.

*

On Microsoft Windows, if you use the special value *, the client library prompts for a new password during a connection attempt only if the existing password has expired. The user must provide their existing password, provide their new password, and confirm their new password. When the user completes the fields and clicks *OK*, the old password is authenticated and the database server attempts to change the password. If you have set the verify_password_function option, the new password is verified. The process of verifying if a user's password has expired, prompting for a password, and authenticating and changing the password occurs with a single connect call to the client library.

Default

The password is not changed, and the client library does not prompt for a new password.

Remarks

If you are authenticating with an Integrated or Kerberos login, the original password is not validated, the database server ignores the new password value and the password is not changed, and access to the database is denied.

If you are authenticating using LDAP user authentication (LDAPUA) or PAM user authentication (PAMUA), this connection parameter has no effect.

This connection parameter is very effective when you implement a login policy using the password_life_time or password_expiry_on_next_login options. Alternatively, you can implement a password expiry policy by having the login_procedure signal the `Password has expired` error.

A user receives a `Password has expired error` if their environment does not support password prompting. In a Microsoft Windows environment, the prompt window might not correctly prevent interaction with the calling application's window (it may not be modal or have the correct parent window) if the calling application has multiple top-level windows or if the application's top level windows are minimized.

In a Windows environment, if you use the ODBC SQLDriverConnect function and the DriverCompletion argument is anything other than SQL_DRIVER_NOPROMPT, the connection prompts for a new password if the

password has expired. The connection might prompt for a new password in OLE DB when the DBPROP_INIT_PROMPT property is anything other than DBPROMPT_NOPROMPT. Both cases function as if the NewPassword=* connection parameter was specified.

Example

The following connection string changes the password of user Test1 when they connect:

```
"UID=Test1;PWD=welcome;NEWPWD=hello"
```

In a Windows environment, the following connection string prompts the user Test1 for a new password when the existing password expires:

```
"UID=Test1;PWD=welcome;NEWPWD=*"
```

Related Information

login_procedure Option [page 755]
verify_password_function Option [page 866]
post_login_procedure Option [page 791]

1.1.3.34 MatView (MATVIEW) Connection Parameter

Specifies the string to return for materialized views when the ODBC function SQLTables is called.

Syntax

```
MATVIEW=string
```

Usage

ODBC, OLE DB

Default

VIEW

Remarks

By default, when you run the ODBC function SQLTables, the value VIEW is returned in the TABLE_TYPE column for materialized views. Use the MatView connection parameter to specify a different return value. The string that you specify is also returned when the SQLTables function is called with a TableType filter.

Example

To have the SQLTables function return materialized views as type MATERIALIZED VIEW, connect as follows:

```
UID=DBA;PWD=passwd;ServerName=server-name;MATVIEW=MATERIALIZED VIEW
```

Related Information

ODBC Support

1.1.3.35 NodeType (NODE) Connection Parameter

Selects which type of server, participating in a mirroring configuration, to connect to.

Used in high-availability and read-only scale-out configurations. Depending on the setting, the connection may be redirected to another server, including providing load balancing.

Syntax

```
{ NodeType | NODE }={ DIRECT | PRIMARY | MIRROR | COPY | READONLY}
```

Usage

Network server only.

Allowed Values

DIRECT

This is the default setting.

When the NodeType connection parameter is set to DIRECT, the database server accepts the connection without performing load balancing or redirection.

PRIMARY

If the NodeType connection parameter is set to PRIMARY and you have connected to the primary server, the connection is accepted. If you have connected to a non-primary server (such as, the mirror server or a copy node), the database server redirects the connection to the primary server.

COPY

When the NodeType connection parameter is set to COPY, the database server performs load balancing and chooses a copy node for the connection. In a read-only scale-out system, the database server examines the copy nodes in its own branch (including itself if it is not the root node) and chooses the copy node with the lightest load. If the database server does not choose itself, it redirects the client to the chosen database server.

MIRROR

If the NodeType connection parameter is set to MIRROR and you have connected to the mirror server, the connection is accepted. If you have connected to a non-mirror server, the database server redirects the connection to the mirror server.

READONLY

When the NodeType connection parameter is set to READONLY, you are connected to any read-only server, either a copy node or the mirror. If there are no copy nodes, this is equivalent to the MIRROR setting. If there are copy nodes, this is equivalent to the COPY setting.

Default

DIRECT

Remarks

This connection parameter is used by client applications to select a server that is performing the specified type of role. The database server that the client initially connects to determines which database server should handle the connection, and if necessary, redirects the connection to the appropriate database server by returning the address of the database server. The client is automatically disconnected from the original database server, and then automatically connected to the appropriate database server.

When you specify COPY, load balancing occurs on the branch that contains the database server the client initially connected to. A branch consists of that database server and any of its children. This feature can be useful for performing load balancing among geographically separate servers.

When NodeType is specified, the application typically connects to the root node, and that database server determines which copy node is least heavily loaded. The connection is then redirected to that copy node. If the application makes and drops several such connections within a short period of time, the connection is pooled and the root node is not asked which copy server to use. Using connection pooling reduces the load on the root node, but may not give expected behavior. The application can specify that its connections cannot be pooled to ensure that the root server determines which copy node to connect on each connection.

If a client connection is redirected to a different database server, the value specified in the ServerName (Server) connection parameter and the value of the Name database server property do not match.

If you are using this parameter to redirect connections that use any of the following user authentication methods, then all the database servers in the scale-out tree must be configured as follows:

Integrated login

(Microsoft Windows only) All of the database servers in the scale-out tree must be a part of the same Microsoft Windows user group or be configured to allow the same Microsoft Windows users to access them.

Kerberos user authentication

All of the database servers in the scale-out tree must have Kerberos user authentication enabled. The Kerberos server principal for all the servers in the scale-out tree must have the same realm.

LDAP user authentication

All of the database servers in the scale-out tree must have LDAP user authentication enabled. To enable TLS encryption for communications with the LDAP server, each database server needs access to the trusted certificates file. Therefore, the trusted certificates file must be stored using the same file path on each computer.

Whenever authentication with the LDAP server succeeds, an attempt is made to update the user's password in the database if the stored value is different. This may succeed or fail depending on whether the database is read-only. Also, the updated password is not replicated to other nodes. So if the password has changed and the LDAP server is unavailable, failover to Standard user authentication may fail.

PAM user authentication

(UNIX and Linux only) All of the database servers in the scale-out tree must have PAM user authentication enabled. Therefore, each computer must have the appropriate PAM configuration defined.

Whenever authentication with the PAM service succeeds, an attempt is made to update the user's password in the database if the stored value is different. This may succeed or fail depending on whether the database is read-only. Also, the updated password is not replicated to other nodes. So if the password has changed and the PAM service is unavailable, failover to Standard user authentication may fail.

Database mirroring and read-only scale-out each require a separate license.

Related Information

Database Mirroring [page 1757]
Read-only Scale-out [page 1830]
Separately Licensed Components
ServerName (Server) Connection Parameter [page 110]

1.1.3.36 Password (PWD) Connection Parameter

Provides a password for a connection.

Syntax

```
{ Password | PWD }=password-string
```

Usage

Anywhere

Allowed Values

password-string

Passwords have a maximum length of 255 bytes and are case sensitive. Passwords cannot include leading spaces, trailing spaces, or semicolons.

Default

No password provided.

Remarks

Every database user has a password. When using Standard authentication, the user supplies a password when connecting to the database. Other forms of authentication do not require a password.

By default, passwords must be 6 bytes in length. To change this requirement, set the min_password_length database option.

An application can include the password as plain text in the connection string by using the Password (PWD) connection parameter. An application can include the password as encoded text in the connection string by using the EncodedPassword (ENP) connection parameter. If both the Password (PWD) connection parameter and the EncodedPassword (ENP) connection parameter are specified, then the Password (PWD) connection parameter takes precedence.

Passwords are encrypted by the client before they are sent to the database server.

⚠ Caution

When creating a data source, do not include the password as part of the definition. Although both the *ODBC Configuration For SQL Anywhere* window in the Windows *ODBC Data Source Administrator* and the

SQL Anywhere Data Source utility (dbdsn) allow you to specify a password,, including this information in a data source poses a security risk.

On Unix, data source information is stored in a system information file (named `.odbc.ini` by default).

Example

The following connection string fragment supplies a user ID and a password.

```
UID=DBA;PWD=passwd
```

Related Information

Password and User ID Restrictions and Considerations [page 1634]
Security: Passwords [page 1685]
Case Sensitivity
Connection Parameters and Connection Strings [page 35]
ODBC Data Sources on UNIX/Linux [page 126]
Userid (UID) Connection Parameter [page 115]
EncodedPassword (ENP) Connection Parameter [page 78]
min_password_length Option [page 772]

1.1.3.37 PrefetchBuffer (PBUF) Connection Parameter

Sets the maximum amount of memory for buffering rows, in bytes.

Syntax

```
{ PrefetchBuffer | PBUF }=buffer-size[ k | m ]
```

Usage

Anywhere

Allowed Values

buffer-size

The value is in bytes, but you can use k or m to specify units of kilobytes or megabytes, respectively. This connection parameter accepts values between 64 KB and 8 MB.

Default

512 KB (524288 bytes)

Remarks

The PrefetchBuffer (PBUF) connection parameter controls the per-connection maximum memory allocated on the client to store prefetched rows.

In some circumstances, increasing the number of prefetched rows can improve query performance. You can increase the number of prefetched rows using the PrefetchRows (PROWS) and PrefetchBuffer (PBUF) connection parameters.

Increasing the PrefetchBuffer (PBUF) connection parameter also increases the amount of memory used to buffer GET DATA requests. This may improve performance for some applications that process many GET DATA (SQLGetData) requests.

Example

The following connection string fragment could be used to determine if the PrefetchBuffer memory limit is reducing the number of prefetched rows.

```
...PrefetchRows=100;LogFile=c:\temp\client.txt
```

The following string could be used to increase the memory limit to 2 MB:

```
...PrefetchRows=100;PrefetchBuffer=2M
```

Related Information

Connection Parameters and Connection Strings [page 35]
PrefetchRows (PROWS) Connection Parameter [page 107]

1.1.3.38 PrefetchOnOpen Connection Parameter

Sends a prefetch request with a cursor open request when this parameter is enabled.

Syntax

```
PrefetchOnOpen={ YES | NO }
```

Usage

ODBC

Default

NO

Remarks

Enabling this option sends a prefetch request with a cursor open request, thereby eliminating a network request to fetch rows each time a cursor is opened. Columns must already be bound in order for the prefetch to occur on the open. Rebinding columns between the cursor open and the first fetch when using PrefetchOnOpen will cause reduced performance.

Making ODBC calls to SQLExecute or SQLExecDirect on a query or stored procedure which returns a result set causes a cursor open.

Enabling this option can improve performance if your:

- network exhibits poor latency.
- application sends many cursor open and close requests.

1.1.3.39 PrefetchRows (PROWS) Connection Parameter

Provides an initial suggested number of rows to prefetch when querying the database.

Syntax

```
{ PrefetchRows | PROWS }=number-of-rows
```

Usage

Anywhere

Allowed Values

number-of-rows

The number of rows prefetched is limited both by the PrefetchRows (PROWS) connection parameter and the PrefetchBuffer (PBUF) connection parameter, which limits the memory available for storing prefetched rows.

The maximum number of rows that can be prefetched is 10000.

Default

10

200 for ADO.NET

Remarks

By default, the client library dynamically increases the number of prefetch rows for cursors that would gain a performance benefit from such an increase. Because of this behavior, increasing the PrefetchRow default does not significantly improve the performance of most applications.

Increasing the PrefetchRow default may improve the performance of ODBC, OLE DB, and SQL Anywhere JDBC applications that use STATIC or FORWARD ONLY FOR UPDATE cursor types but do mostly fetch next operations and very rare positioned updates or deletes, particularly if the client and database server are communicating over a slow or wide area network.

The number of prefetched rows is limited by the PrefetchBuffer (PBUF) connection parameter, which limits the memory available for storing prefetched rows.

The maximum number of rows that can be prefetched is 10000.

Example

The following connection string fragment sets the number of prefetched rows to 100:

```
...PrefetchRows=100;...
```

Related Information

Connection Parameters and Connection Strings [page 35]
PrefetchBuffer (PBUF) Connection Parameter [page 105]

1.1.3.40 RetryConnectionTimeout (RetryConnTO) Connection Parameter

Instructs the client library (DBLIB, ODBC, ADO, and so on) to keep retrying the connection attempt, as long as the server is not found, for the specified period of time.

Syntax

```
{ RetryConnectionTimeout | RetryConnTO }=timeout-value
```

Usage

Anywhere

Allowed Values

timeout-value

The value specified by this connection is a timeout, in seconds. It is not a counter of the number of times to retry the connection attempt. The default value of zero indicates that the connection attempt should only be tried once.

Default

0

Remarks

There is a half-second delay between iterations, and the retries only occur if the connection attempt failed because the database server was not found. Any other error is returned immediately. If the database server is not found, the connection attempt will take at least as long as the time specified by the RetryConnectionTimeout connection parameter.

The default TCP timeout is 5 seconds, so if your connection string contains a value for RetryConnTO that is less than 5, for example `Host=host-name;RetryConnTO=3`, then the connection attempt still takes 5 seconds.

Example

The following connection string fragment tells the client library to continue to retry the connection attempt for at least 5 seconds:

```
...RetryConnTO=5;...
```

Related Information

Timeout (TO) Protocol Option [page 242]

1.1.3.41 ServerName (Server) Connection Parameter

Specifies the name of a running database server to which you want to connect.

Syntax

```
{ ServerName | Server }=database-server-name-string
```

Usage

Anywhere

Allowed Values

database-server-name-string

If you are starting a database server automatically, you can provide a server name using this parameter.

The server name is interpreted according to the character set of the client computer. Non-ASCII characters are not recommended in server names.

Names must be valid identifiers. Database server names cannot:

- begin with white space, single quotes, or double quotes

- end with white space
- contain semicolons, forward slashes (/), or backslashes (\)
- be longer than 250 bytes
- contain spaces when they are running on UNIX or Linux

Long database server names are truncated to different lengths depending on the protocol.

On Windows and UNIX/Linux, version 9.0.2 and earlier clients cannot connect to version 10.0.0 and later database servers with names longer than the following lengths:

- 40 bytes for Windows shared memory
- 31 bytes for UNIX and Linux shared memory
- 40 bytes for TCP/IP

Default

The default local database server.

Remarks

EngineName and ENG are accepted for backward compatibility, but are deprecated.

When a database server starts, it attempts to become the default database server on that computer. The first database server to start when there is no default server becomes the default database server. Each Windows session has its own default database server. Shared memory connection attempts on that computer that do not explicitly specify a database server name connect to the default server.

ServerName is not needed to connect to the default local database server.

If you use the Host connection parameter, ServerName is not required if both the host name and port are provided or if there is only one server running on the host using the default port.

In the *Connect* window, and in the *ODBC Configuration For SQL Anywhere* window, this is the *Server Name* field.

> **i** Note
>
> It is recommended that you include the ServerName parameter in connection strings for deployed applications. This ensures that the application connects to the correct server in the case where a computer is running multiple database servers, and can help prevent timing-dependent connection failures.
>
> Use the -xd option for database servers being used by deployed applications to prevent the database server from becoming the default database server. All clients should explicitly specify the name of the database server to which they must connect by using the ServerName connection parameter. This ensures that the application connects to the correct database server when a host is running multiple database servers.

Example

Connect to a server named Guelph:

```
Server=Guelph
```

Related Information

Connection Parameters and Connection Strings [page 35]
Connection Parameters to Use with Embedded Databases [page 42]
Identifiers
-n Database Server Option [page 467]
-xd Database Server Option [page 526]

1.1.3.42 StartLine (START) Connection Parameter

Starts a local database server running from an application.

Syntax

```
{ StartLine | START }=local-database-server-command
```

Usage

Local database servers

Allowed Values

local-database-server-command

By default, the client attempts to connect to a running database server. If no server can be found with the specified connection parameters, the client automatically starts a new local database server using the `local-database-server-command`. The database server is not started automatically if the Host connection parameter is specified or the CommLinks (LINKS) parameter includes TCPIP.

Default

dbeng17

Remarks

The StartLine connection parameter is only used to start a database server if a connection cannot be made to the specified database server.

For example, suppose you start a database server running a database as follows:

```
dbeng17 c:\mydb.db
```

Connect another database (without specifying a database server name using the ServerName connection parameter):

```
dbisql -c "START=dbsrv17 -x none -c 8M;DBN=seconddb;DBF=c:
\myseconddb.db;UID=DBA;PWD=passwd"
```

In this case, the dbsrv17 database server is not started. Instead, the dbeng17 database server that was used to start `mydb.db` is used to start and connect to `myseconddb.db`.

However, if Server=`server-name` had been specified, and a database server named `server-name` was not running, then the dbsrv17 database server would have started.

> **i** Note
>
> To specify the database name, database file, or server name, it is recommended that you use the DatabaseName (DBN), DatabaseFile (DBF), and ServerName(Server) connection parameters, rather than the specifying database server options in the StartLine connection parameter.
>
> The following command uses the recommended syntax:
>
> ```
> START=dbsrv17 -c 8M;Server=myserver;DBF=c:\sample.db;DBN=mydb
> ```
>
> The following syntax is not recommended:
>
> ```
> START=dbsrv17 -c 8M -n myserver "c:\sample.db"
> ```

Example

The following data source fragment starts a personal database server with a cache of 8 MB.

```
"DBF=C:\Users\Public\Documents\SQL Anywhere
         17\Samples\demo.db;StartLine=dbeng17 -c 8M"
```

Related Information

Connection Parameters and Connection Strings [page 35]
Connection Parameters to Use with Embedded Databases [page 42]
CommLinks (LINKS) Connection Parameter [page 59]
SQL Anywhere Database Server Executable (dbsrv17, dbeng17) [page 380]

1.1.3.43 Unconditional (UNC) Connection Parameter

Stops a database server using the db_stop_engine function, or a database using the db_stop_database function, even when there are connections to the database server.

Syntax

```
{ Unconditional | UNC }={ YES | NO }
```

Usage

db_stop_engine and db_stop_database functions only.

Default

NO

Remarks

The db_stop_engine and db_stop_database functions shut down a database server or database, respectively. If you specify UNC=YES in the connection string that is passed to the db_stop_database function or the db_stop_engine function, then database server or database is shut down even if there are active connections. If UNC is not set to YES, then the database server or database is shut down only if there are no active connections.

Related Information

Connection Parameters and Connection Strings [page 35]
db_stop_database Function

db_stop_engine Function

1.1.3.44 Userid (UID) Connection Parameter

Specifies the user ID used to log in to the database.

Syntax

```
{ Userid | UID }=userid
```

Usage

Anywhere

Allowed Values

userid

User IDs cannot:

- begin with white space, single quotes, or double quotes
- end with white space
- contain semicolons

Default

None

Remarks

You must always supply a user ID when connecting to a database, unless you are using an integrated or Kerberos login.

Example

The following connection string fragment supplies the user ID DBA and password passwd:

```
UID=DBA;PWD=passwd
```

Related Information

Connection Parameters and Connection Strings [page 35]
User Security (Roles and Privileges) [page 1526]
Password (PWD) Connection Parameter [page 104]

1.1.4 ODBC Data Sources

Microsoft **Open Database Connectivity** (**ODBC**) is a standard application programming interface for connecting client applications to Windows-based database management systems.

Many client applications, including application development systems, use the ODBC interface to access databases. When connecting to the database, ODBC applications typically use ODBC data sources. Many other types of applications support the use of ODBC data sources (for example, OLE DB or .NET).

You cannot use ODBC data sources with SAP Open Client or jConnect.

The SQL Anywhere ODBC driver is named *SQL Anywhere 17*.

An ODBC data source is a set of connection parameters, stored in the registry or in a file.

> ⚠ Caution
>
> Storing user IDs, encrypted or unencrypted passwords, and database keys in a data source is not recommended.

In this section:

Creating an ODBC File Data Source (ODBC Data Source Administrator) [page 125]
Create an ODBC file data source in ODBC Data Source Administrator.

ODBC Data Sources on UNIX/Linux [page 126]
On UNIX and Linux operating systems, ODBC data sources are held in a system information file.

Related Information

ODBC Conformance
Security: Passwords [page 1685]

1.1.4.1 Creating an ODBC Data Source (ODBC Data Source Administrator)

Create an ODBC data source using the ODBC Data Source Administrator.

Context

On Windows, system data sources are available to all users of the computer system and to Windows services. User data sources are available only to the user for which it was created.

> **i** Note
>
> **Creating a system data source on 64-bit Windows**
>
> System data sources are available to all users of the computer system and to Windows services. 64-bit versions of Windows maintain two sets of system data sources in the Windows Registry: one for 64-bit client applications and one for 32-bit client applications. To create a system data source that is accessible to 32-bit applications, you must run the 32-bit ODBC Data Source Administrator (located in the `%windir%\SysWOW64` folder). To avoid connection problems, it is recommended that you create both 64-bit and 32-bit system data sources and ensure that they are configured identically.
>
> **Creating a user data source on 64-bit Windows**
>
> User data source definitions are stored in the part of the Windows Registry containing settings for the user currently logged on to the system. 64-bit versions of Windows maintain one set of user data sources in the Windows Registry for each user of the computer. You can use either the 32-bit or the 64-bit version of the ODBC Data Source Administrator to create a user data source. This data source can be used by both 64-bit and 32-bit client applications as long as the matching ODBC driver is installed.

Procedure

1. For the 32-bit ODBC Data Source Administrator, click *Start* ▸ *Programs* ▸ *SQL Anywhere 17* ▸ *Administration Tools* ▸ *ODBC Data Source Administrator (32-bit)*.

   For the 64-bit ODBC Data Source Administrator, click *Start* ▸ *Programs* ▸ *SQL Anywhere 17* ▸ *Administration Tools* ▸ *ODBC Data Source Administrator (64-bit)*.
2. To create an ODBC data source for the current user, click the *User DSN* tab.

   To create a system data source, click the *System DSN* tab.
3. Click *Add*.
4. In the *Name* list, choose *SQL Anywhere 17*. Click *Finish*.
5. Specify the connection parameters for the ODBC data source.

   > ⚠ Caution
   >
   > Storing user IDs, encrypted or unencrypted passwords, and database keys in a data source is not recommended.
6. Click *OK*.
7. Click *OK* to close the *ODBC Data Source Administrator*.

Results

The ODBC data source is created.

If you are creating a system data source on 64-bit Windows, make sure to create both 32-bit and 64-bit system data sources using the same connection parameters.

1.1.4.2 Editing ODBC Data Sources (ODBC Data Source Administrator)

Edit an ODBC data source.

Prerequisites

An ODBC data source must be defined.

Context

> **Note**
>
> **Editing a system data source on 64-bit Windows**
>
> System data sources are available to all users of the computer system and to Windows services. 64-bit versions of Windows maintain two sets of system data sources in the Windows Registry: one for 64-bit client applications and one for 32-bit client applications. To edit a system data source that is accessible to 32-bit applications, you must run the 32-bit ODBC Data Source Administrator (located in the `%windir%\SysWOW64` folder). To avoid connection problems, it is recommended that you maintain both 64-bit and 32-bit system data sources and ensure that they are configured identically.
>
> **Editing a user data source on 64-bit Windows**
>
> User data source definitions are stored in the part of the Windows Registry containing settings for the user currently logged on to the system. 64-bit versions of Windows maintain one set of user data sources in the Windows Registry for each user of the computer. You can use either the 32-bit or the 64-bit version of the ODBC Data Source Administrator to edit a user data source. This data source can be used by both 64-bit and 32-bit client applications as long as the matching ODBC driver is installed.

Procedure

1. For the 32-bit ODBC Data Source Administrator, click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *ODBC Data Source Administrator (32-bit)*.

   For the 64-bit ODBC Data Source Administrator, click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *ODBC Data Source Administrator (64-bit)*.
2. Click one of the *User DSN* or *System DSN* tabs.
3. In the *Name* list, click a data source.
4. Click *Configure*.
5. Edit the connection parameters for the ODBC data source.

   > **Caution**
   >
   > Storing user IDs, encrypted or unencrypted passwords, and database keys in a data source is not recommended.
6. Click *OK*.
7. Click *OK* to close the *ODBC Data Source Administrator*.

Results

The ODBC data source is edited and saved.

If you are updating a system data source on 64-bit Windows, make sure to apply the same updates to both 32-bit and 64-bit system data sources.

1.1.4.3 Creating an ODBC Data Source (dbdsn Utility)

Create an ODBC data source.

Context

On Windows, system data sources are available to all users of the computer system and to Windows services. User data sources are available only to the user for which it was created. File data sources cannot be created with the dbdsn utility. Use the ODBC Data Source Administrator to create file data sources.

On Linux and other non-Windows systems, the dbdsn utility can be used to create ODBC data sources.

> ℹ Note
>
> **Creating a system data source on 64-bit Windows**
>
> System data sources are available to all users of the computer system and to Windows services. 64-bit versions of Windows maintain two sets of system data sources in the Windows Registry: one for 64-bit client applications and one for 32-bit client applications. To create a system data source that is accessible to 32-bit applications, you must run the 32-bit dbdsn utility (located in the bin32 subdirectory of your SQL Anywhere installation directory). To avoid connection problems, it is recommended that you create both 64-bit and 32-bit system data sources and ensure that they are configured identically.
>
> **Creating a user data source on 64-bit Windows**
>
> User data source definitions are stored in the part of the Windows Registry containing settings for the user currently logged on to the system. 64-bit versions of Windows maintain one set of user data sources in the Windows Registry for each user of the computer. You can use either the 32-bit or the 64-bit version of the dbdsn utility to create a user data source. This data source can be used by both 64-bit and 32-bit client applications as long as the matching ODBC driver is installed.

> ⚠ Caution
>
> Storing user IDs, encrypted or unencrypted passwords, and database keys in a data source is not recommended.

Procedure

Run the Data Source utility (dbdsn) from a command prompt.

Results

The ODBC data source is created.

Example

The following command creates an ODBC data source for the sample database. The command must be entered on one line:

```
dbdsn -w "My DSN" -c "DBF=$SQLANYSAMP17/demo.db"
```

Related Information

File Data Sources on Microsoft Windows [page 124]
Data Source Utility (dbdsn) [page 1140]

1.1.4.4 Creating an ODBC Data Source on macOS

Create an ODBC data source using the ODBC Administrator Tool.

Prerequisites

Your macOS system must not be version 10.6 or later.

The SQL Anywhere ODBC driver must be installed.

The ODBC Administrator Tool for macOS must be installed. This tool is available from Apple.

The ODBC Administrator Tool for macOS must be configured to use the SQL Anywhere ODBC driver first. Once this is done, then you may create data sources that use this driver.

Context

Use of the ODBC Administrator Tool requires that either the ODBCINI or ODBC_INI environment variable be used to specify the non-standard location and file name of the ODBC initialization file (`~/Library/ODBC/odbc.ini`).

To avoid this issue, you can use the Data Source utility (dbdsn) to create ODBC data sources on macOS.

Procedure

1. Launch the ODBC Administrator Tool for macOS from `/Applications/Utilities`.
2. Click the *User DSN* tab, and then click *Add*.
3. In the *Name* list, click *SQL Anywhere 17*.
4. Click *OK*.
5. In the *Data Source Name* field, type a name for your data source.
6. In the *Description* field, type a suitable description for your data source.
7. Specify valid connection parameters and values. Some values may be case insensitive. The following is an example.

Keyword	Value
DatabaseFile	/Applications/SQLAnywhere17/samples/demo.db
StartLine	dbeng17
ServerName	demo17
ThreadManager	ON

> ⚠ Caution
>
> Storing user IDs, encrypted or unencrypted passwords, and database keys in a data source is not recommended.

8. Click *OK*.
9. Click *Apply*.
10. Press Command+Q to exit.

Results

The ODBC data source is created.

In this section:

Configure the ODBC Administrator Tool for macOS [page 123]
Add the SQL Anywhere ODBC driver to the ODBC Administrator Tool for macOS so that you can create an ODBC data source that uses this driver.

Related Information

ODBC Data Sources on UNIX/Linux [page 126]
Alphabetical List of Connection Parameters [page 45]
Data Source Utility (dbdsn) [page 1140]

1.1.4.4.1 Configure the ODBC Administrator Tool for macOS

Add the SQL Anywhere ODBC driver to the ODBC Administrator Tool for macOS so that you can create an ODBC data source that uses this driver.

Prerequisites

Your macOS system must not be version 10.6 or later.

The SQL Anywhere ODBC driver must be installed.

The ODBC Administrator Tool for macOS must be installed. This tool is available from Apple.

Context

You can also use the Data Source utility (dbdsn) to create ODBC data sources on macOS.

Procedure

1. Launch the ODBC Administrator Tool for macOS from `/Applications/Utilities`.
2. Click the *Drivers* tab.
3. Click *Add*.
4. In the *Description* field, type ***SQL Anywhere 17***.
5. Click *Choose* and select the SQL Anywhere ODBC driver in both the *Driver File Name* and *Setup File Name* fields. By default, it is located in */Applications/SQLAnywhere17/System*`/lib64/dbodbc`*17*`_r.bundle`.

   The `_r` in the bundle name indicates that it is the threaded version of the driver. There is also an unthreaded version (`dbodbc`*17*`.bundle`) for use with unthreaded applications.
6. Click *OK*.

   The ODBC configuration files are located in `/Library/ODBC` within your home directory. There is an `odbcinst.ini` file for driver information and an `odbc.ini` file for data source information. You can edit the information with a text editor.

   You must use either the ODBCINI or ODBC_INI environment variable to specify the non-standard location and file name of the ODBC initialization file (`~/Library/ODBC/odbc.ini`).

Results

The SQL Anywhere ODBC driver is added.

Related Information

Creating an ODBC Data Source (dbdsn Utility) [page 120]

1.1.4.5 File Data Sources on Microsoft Windows

File data sources, which are stored as files, are an alternative to ODBC data sources that are stored in the system registry.

On Microsoft Windows, file data sources typically have the extension `.dsn`. They consist of sections, and each section starts with a name enclosed in square brackets.

To connect using a file data source, use the FileDataSourceName (FILEDSN) connection parameter. You cannot use both DataSourceName (DSN) and FileDataSourceName (FILEDSN) in the same connection string.

Before you create or modify a FileDSN, the database server must be running and the database must be started. A connection is always attempted by the Microsoft ODBC Data Source Administrator and if the connection is not successful, one of two things happens:

For a new file data source

The Microsoft ODBC Data Source Administrator displays a message stating that the specified file data source parameters could not be used to establish a connection. The message then asks whether or not you want to save the non-verified file. If you choose to save the file data source, the ODBC Data Source Administrator writes only the following lines to the file:

```
[ODBC]
DRIVER=SQL Anywhere 17
```

For an existing file data source

The Microsoft ODBC Data Source Administrator displays a message stating that the specified file data source name is invalid. The file data source is not updated by the ODBC Data Source Administrator.

If the connection is successful, the new or updated file data source is written to disk by the ODBC Data Source Administrator, but the Password (PWD) connection parameter is not included (and is removed if it was previously present in the file). The same is not true of the EncodedPassword (ENP) connection parameter. It is preserved.

> **i** Note
>
> You can use file data sources to distribute the file to users and simplify the management of multiple user connections. If the file is placed in the default location for file data sources, it is picked up automatically by ODBC.

1.1.4.6 Creating an ODBC File Data Source (ODBC Data Source Administrator)

Create an ODBC file data source in ODBC Data Source Administrator.

Prerequisites

To successfully create a file data source, you must be able to establish a connection to the database for which you are creating the file data source.

Context

> ⚠ Caution
>
> Storing user IDs, encrypted or unencrypted passwords, and database keys in a data source is not recommended.

Procedure

1. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *ODBC Data Source Administrator*.
2. Click the *File DSN* tab.
3. Click *Add*.
4. On the *Driver* list, click *SQL Anywhere 17*.
5. Click *Next*.
6. Follow the instructions in the *Create New Data Source Wizard*.
7. Click *OK*.
8. Click *OK* to close the *ODBC Data Source Administrator*.

Results

The ODBC data source file is created.

1.1.4.7 ODBC Data Sources on UNIX/Linux

On UNIX and Linux operating systems, ODBC data sources are held in a system information file.

This file is usually named `.odbc.ini`. The database server searches the following locations, in order, for the system information file:

- The ODBCINI environment variable.
- The ODBC_INI environment variable.
- The ODBCHOME environment variable.
- The HOME environment variable.
- The user's home directory (~).
- The PATH environment variable.

> **i Note**
>
> The ODBCINI and ODBC_INI environment variables can be used to locate the system information file (which is usually named `.odbc.ini`), while the ODBCHOME and HOME environment variables can be used to define the path where the `.odbc.ini` file is located.
>
> Both ODBCINI and ODBC_INI specify a full path, including the file name. If the system information file is located in a directory specified by ODBCINI or ODBC_INI, it does not have to be named `.odbc.ini`.

The following is a sample system information file:

```
[My Data Source]
Host=hostname
ServerName=myserver
```

You can enter any connection parameter in the system information file.

> **⚠ Caution**
>
> Storing user IDs, encrypted or unencrypted passwords, and database keys in a data source is not recommended.

On UNIX and Linux, use the dbdsn utility to create and manage ODBC data sources.

> **⚠ Caution**
>
> On UNIX and Linux, do not encrypt the system information file (named `.odbc.ini` by default) using the File Hiding utility (dbfhide) unless you are using SQL Anywhere data sources only. If you plan to use other data sources, then encrypting the contents of the system information file may prevent other drivers from functioning properly.
>
> Also, the ODBC initialization file (named `.odbc.ini` by default) may contain sensitive information. Ensure that permissions on the file are as restrictive as possible. For example, consider performing a *chmod 600 .odbc.ini*' on the file.

Related Information

Alphabetical List of Connection Parameters [page 45]
Data Source Utility (dbdsn) [page 1140]
ODBCHOME Environment Variable [UNIX/Linux] [page 572]
ODBCINI and ODBC_INI Environment Variables [UNIX/Linux] [page 573]

1.1.5 Authentication Mechanisms

You can set up your database to allow users to connect using an external authentication mechanism such as LDAP, PAM, Kerberos, or Microsoft Windows integrated login.

The standard or default method to connect to a database is to provide a user ID and password that is known to the database.

In addition to standard authentication, the database server supports the following authentication mechanisms:

- Microsoft Windows integrated login
- LDAP user authentication
- PAM user authentication
- Kerberos user authentication

These authentication mechanisms are convenient for users and administrators as they permit a single security system for database, operating system, and network security. Other advantages include:

- The user does not need to provide the database user ID and password to connect to the database.
- The authenticated user is mapped to a database user ID so the user name and password do not have to match the database user ID and password.
- Multiple users can be mapped to a single database user ID.
- Some authentication mechanisms permit entire groups to be mapped to a single database user ID.

These authentication mechanisms offer the convenience of a single security system, but there are important security concerns that database administrators should be familiar with, such as unrestricted database access and copies of database files. Ensure that you are aware of these concerns and how to address them.

In this section:

Login Modes and the login_mode Option [page 128]
The login_mode option is used to specify the allowed set of user authentication mechanisms.

Microsoft Windows Integrated Login [page 129]
The Windows **integrated login** feature allows you to maintain a single user ID and password for operating system and network logins, and database connections.

LDAP User Authentication [page 139]
Lightweight Directory Access Protocol (LDAP) is an industry standard for accessing directory services.

Tutorial: Creating an LDAP User Authentication Environment [page 144]
This tutorial guides you through the basic steps for setting up an LDAP user authentication environment in Interactive SQL or SQL Central.

PAM User Authentication [page 156]
Pluggable Authentication Module (PAM) is an authentication mechanism that provides a common interface for multiple low-level authentication schemes, allowing a user to use the same credentials on all of their systems.

Kerberos User Authentication [page 159]
The Kerberos login feature allows you to maintain a single user ID and password for database connections, operating system, and network logins.

Security: Use Login Modes to Secure the Database [page 171]
There are several measures you can take to secure your database.

Security: Integrated Logins Can Result in Unrestricted Database Access [page 172]
The Integrated login feature uses native Windows authentication to control access to a database.

1.1.5.1 Login Modes and the login_mode Option

The login_mode option is used to specify the allowed set of user authentication mechanisms.

As database options apply only to the database in which they are found, different databases can have a different login settings even if they are loaded and running on the same server.

The login_mode database option accepts the following values:

Standard

Standard user authentication is permitted. This is the default setting. A database user ID and password must be provided.

Integrated

Windows integrated logins are permitted.

Kerberos

Kerberos logins are permitted.

LDAPUA

LDAP (Lightweight Directory Access Protocol) user authenticated logins are permitted.

PAMUA

PAM (Pluggable Authentication Modules) user authenticated logins are permitted.

CloudAdmin

This login mode is for internal use by the cloud.

> ⚠ Caution
>
> Setting the login_mode database option to not allow Standard user authentication (user ID and password) restricts connections to only those users or groups who have been granted an Integrated, Kerberos, LDAPUA, PAMUA, or CloudAdmin login mapping. Attempting to connect with a user ID and password generates an error unless you are a user with the MANAGE ANY USER privilege.

To allow more than one type of login, specify multiple values for the login_mode option. For example, the following SQL statement sets the value of the login_mode database option to allow both standard and integrated logins:

```
SET OPTION PUBLIC.login_mode = 'Standard,Integrated';
```

If a database file can be copied, the temporary public login_mode option should be used (for login modes other than standard). In this way, login modes other than standard are not supported by default if the file is copied.

```
SET TEMPORARY OPTION PUBLIC.login_mode = 'Standard,Integrated';
```

Related Information

Kerberos User Authentication [page 159]
login_mode Option [page 752]

1.1.5.2 Microsoft Windows Integrated Login

The Windows **integrated login** feature allows you to maintain a single user ID and password for operating system and network logins, and database connections.

To create an integrated login:

- Enable the integrated login feature.
- Create a database user to map the integrated login to (if one does not already exist).
- Create an integrated login mapping between a Windows user or group profile and an existing database user. The *Login Mappings* folder in SQL Central lists all users with integrated login privileges.
- Connect from a client application and test the integrated login facility.

Supported Operating Systems

Integrated login capabilities are available for Windows-based database servers. Windows clients can use integrated logins to connect to a network server running on Windows.

Windows will only return the names of global groups in which the user is a direct member, or the names of local groups containing global groups in which the user is a direct member.

If user `userA` is listed in global group `groupB` which is, in turn, listed in global group `groupC`, then only `groupB` is returned. Global group `groupC` is not returned even though it contains global group `groupB`.

If the database server's local group `localD` contains `groupB`, then `userA` is located by indirection in `localD`.

If the database server's local group `localD` contains `groupC`, then `userA` is not located by indirection in `localD`.

Integrated Login Benefits

An integrated login is a mapping from one or more Windows users or Windows user group profiles to an existing database user. A user who has successfully navigated the security for that user profile or group and logged in to a computer can connect to a database without providing an additional user ID or password.

To do this, the database must be configured to use integrated logins and a mapping must have been granted between the user or group profile used to log in to the computer or network, and a database user.

Using an integrated login is more convenient for the user and permits a single security system for database and network security. The advantages of an integrated login include:

- Users do not need to type a user ID or password.
- Users are authenticated by the operating system. A single system is used for database security and computer or network security.
- Multiple user or group profiles can be mapped to a single database user ID.
- The name and password used to log in to the Windows computer do not have to match the database user ID and password.

> ⚠ Caution
>
> Integrated logins offer the convenience of a single security system, but there are important security concerns that database administrators should be familiar with, such as unrestricted database access and copies of database files. Ensure that you are aware of these concerns and how to address them.

In this section:

Related Information

Security: Use Login Modes to Secure the Database [page 171]

1.1.5.2.1 Creating a Windows Integrated Login Mapping (SQL Central)

Use a Windows integrated login to maintain a single user ID and password for operating system and network logins, as well as your database connections.

Prerequisites

You must have the MANAGE ANY USER system privilege and either the SELECT ANY TABLE system privilege or the SELECT privilege on the SYSLOGINMAP system view.

If you need to change the login mode, you must also have the SET ANY SECURITY OPTION system privilege.

Context

A user or group profile does not have to exist for it to be mapped to a database user ID. More than one user profile can be mapped to the same database user ID.

User profiles can only be mapped to an existing database user ID. When that database user ID is removed from the database, all integrated login mappings based on that database user ID are automatically removed.

Procedure

1. Use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the *Folders* view, right-click *Login Mappings* and click *New* > *Login Mapping*.
3. Select *Integrated login mapping* and click *Next*.
4. In the *Which Windows user will be connecting to the database* field, type the name of the user or group profile for whom the integrated login is to be created.
5. When prompted to specify the login mode, select *Standard* and *Integrated*.
6. In the *Which database user do you want to associate with the Windows user* list, select the database user ID this user maps to.
7. Follow the remaining instructions in the *Create Login Mapping Wizard*.

Results

The integrated login mapping is created.

1.1.5.2.2 Creating an Integrated Login (SQL)

Use a Windows integrated login to maintain a single user ID and password for operating system and network logins, as well as your database connections.

Prerequisites

You must have the MANAGE ANY USER system privilege and either the SELECT ANY TABLE system privilege or the SELECT privilege on the SYSLOGINMAP system view.

If you need to change the login mode, you must also have the SET ANY SECURITY OPTION system privilege.

Context

A user or group profile does not have to exist for it to be mapped to a database user ID. More than one user profile can be mapped to the same database user ID.

Procedure

1. Connect to the database.
2. Execute a SET OPTION statement to change the login mode to enable Windows Integrated user authentication.

   For example, the following statement enables both standard and Integrated user authentication.

   ```
   SET OPTION PUBLIC.login_mode='Standard,Integrated';
   ```

   For additional security, use SET TEMPORARY OPTION instead of the SET OPTION statement to set the login_mode option.

   ```
   SET TEMPORARY OPTION PUBLIC.login_mode='Standard,Integrated';
   ```

   The current setting of the login_mode option can be queried as follows:

   ```
   SELECT connection_property('login_mode');
   ```
3. Execute a GRANT INTEGRATED LOGIN TO statement.

Results

The integrated login is created.

Example

The following SQL statement allows Windows users fran_whitney and matthew_cobb to log in to the database as the user DBA without having to know or provide the DBA user ID or password.

```
GRANT INTEGRATED LOGIN
TO fran_whitney, matthew_cobb
AS USER DBA;
```

The following SQL statement allows Windows users who are members of the Windows group mywindowsusers to log in to the database as the user DBA without having to know or provide the DBA user ID or password.

```
GRANT INTEGRATED LOGIN
TO mywindowsusers
AS USER DBA;
```

Related Information

Integrated Logins for Microsoft Windows User Groups [page 136]
GRANT CONNECT Statement

1.1.5.2.3 Revoking an Integrated Login Privilege (SQL Central)

Revoke an integrated login privilege using SQL Central.

Prerequisites

You must have the MANAGE ANY USER system privilege and either the SELECT ANY TABLE system privilege or the SELECT privilege on the SYSLOGINMAP system view.

Procedure

1. Use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *Login Mappings*.
3. In the right pane, right-click the login mapping you want to remove and click *Delete*.
4. Click *Yes*.

Results

The integrate login privilege is revoked.

Related Information

REVOKE Statement

1.1.5.2.4 Revoking an Integrated Login Privilege (SQL)

Revoke an integrated login privilege using the REVOKE statement.

Prerequisites

You must have the MANAGE ANY USER system privilege.

Procedure

1. Connect to the database.
2. Execute a REVOKE INTEGRATED LOGIN FROM statement.

Results

The integrate login privilege is revoked.

Example

The following SQL statement removes integrated login privilege from the Windows user pchin.

```
REVOKE INTEGRATED LOGIN
FROM pchin;
```

Related Information

REVOKE Statement

1.1.5.2.5 Integrated Login Connections from a Client Application

There are two methods you can you to connect using an integrated login:

Set the Integrated (INT) parameter in the list of connection parameters to YES.

If the Integrated (INT) parameter is set to YES in the connection string, an integrated login is attempted. The server attempts a standard login when the connection attempt fails and the login_mode database option includes Standard.

Do not specify a user ID or password in the connection string or Connect window.

If an attempt to connect to a database is made without providing a user ID or password, an integrated login is attempted. The success of the login attempt is dependent on whether the current user profile name matches an integrated login mapping in the database.

Interactive SQL Examples

In the following example, the connection attempt succeeds when the user logs in with a user profile that matches the integrated login mapping in the default database server:

```
CONNECT USING 'INTEGRATED=yes';
```

The CONNECT statement can connect to a database when:

- A server is currently running.
- The default database has the login_mode database option set to accept integrated login connections.
- An integrated login mapping has been created that matches the current user's user profile name or for a Windows user group to which the user belongs.
- A user clicks *OK* without providing more information when the more connection information prompt appears.

Related Information

login_mode Option [page 752]

1.1.5.2.6 Integrated Logins for Microsoft Windows User Groups

You can integrate logins for Microsoft Windows user groups.

When a Windows user logs in, if they do not have an explicit integrated login mapping, but belong to a Windows user group for which there is an integrated login mapping, the user connects to the database as the database user or group specified in the Windows user group's integrated login mapping.

> ⚠ Caution
>
> Creating an integrated login for a Windows user group allows any user that is a member of the group to connect to the database without knowing a user ID or password.

Members of Multiple Groups

If the Windows user belongs to more than one Windows user group, and more than one Windows user group on the computer has an integrated login mapping in the database, then the integrated login only succeeds if all the Windows user groups on the computer have integrated login mappings to the same database user ID. If multiple Windows user groups have integrated login mappings to different database user IDs, an error is returned and the integrated login fails.

For example, consider a database with two user IDs, dbuserA and dbuserB, and the Windows user windowsuser who belongs to the Windows user groups xpgroupA and xpgroupB.

This SQL statement...	Allows...
`GRANT INTEGRATED LOGIN TO windowsuser AS USER dbuserA;`	windowsuser to connect to the database using the integrated login mapping set explicitly for windowsuser.
`GRANT INTEGRATED LOGIN TO xpgroupA AS USER dbuserB;`	windowsuser to connect to the database using the integrated login mapping granted to xpgroupA.
`GRANT INTEGRATED LOGIN TO xpgroupA AS USER dbuserB;` `GRANT INTEGRATED LOGIN TO xpgroupB AS USER dbuserB;`	windowsuser to connect to the database because both Windows user groups that windowsuser belongs to have an integrated login mapping to the same database user.

This SQL statement...	Allows...
`GRANT INTEGRATED LOGIN TO xpgroupA AS USER dbuserA; GRANT INTEGRATED LOGIN TO xpgroupB AS USER dbuserB;`	No connection to the database. When windowsuser attempts to connect to the database, the integrated login fails because each Windows user group has an integrated login mapping to a different database user and windowsuser is a member of both Windows user groups.

Domain Controller Locations

By default, the computer on which the database server is running is used to verify Windows user group membership. If the Domain Controller server is on a different computer than the database server, you can specify the name of the Domain Controller server using the integrated_server_name option. For example:

```
SET OPTION PUBLIC.integrated_server_name = '\\\\myserver-1';
```

Related Information

How to Prevent Microsoft Windows User Groups Members from Connecting to a Database [page 137]
integrated_server_name Option [page 745]

1.1.5.2.7 How to Prevent Microsoft Windows User Groups Members from Connecting to a Database

There are two methods you can use to prevent a user who is a member of a Microsoft Windows user group with an integrated login from connecting to a database using the group integrated login.

- Create an integrated login for the user to a database user ID that does not have a password.
- Create a stored procedure that is called by the login_procedure option to check whether a user is allowed to log in, and raise an exception when a disallowed user tries to connect.

Creating an Integrated Login to a User ID with no Password

When a user is a member of a Windows user group that has an integrated login, but also has an explicit integrated login for their user ID, the user's integrated login is used to connect to the database. To prevent a user from connecting to a database using their Windows user group integrated login, you can create an integrated login for the Windows user to a database user ID without a password. Database user IDs that do not have a password cannot connect to a database.

You can create an integrated login to a user ID with no password in SQL.

1. Add a user to the database without a password. For example:

```
CREATE USER db_user_no_password;
```

2. Create an integrated login for the Windows user that maps to the database user without a password. For example:

```
GRANT INTEGRATED LOGIN TO WindowsUser
AS USER db_user_no_password;
```

Creating a Procedure to Prevent Microsoft Windows Users from Connecting

The login_procedure option specifies the stored procedure to call each time a connection to the database is attempted. By default, the dbo.sp_login_environment procedure is called. You can set the login_procedure option to call a procedure you have written that prevents specific users from connecting to the database.

The following example creates a procedure named login_check that is called by the login_procedure option. The login_check procedure checks the supplied OS user name against a list of users that are not allowed to connect to the database. If the supplied user name is found in the list, the connection fails. In this example, users named Joe, Harry, or Martha are not allowed to connect. If the user is not found in the list, the database connection proceeds as usual and calls the sp_login_environment procedure.

```
CREATE PROCEDURE DBA.login_check()
   BEGIN
      DECLARE INVALID_LOGON EXCEPTION FOR SQLSTATE '28000';
      // Disallow certain users
      IF( connection_property( 'OSUser' ) IN ('Joe','Harry','Martha') ) THEN
        SIGNAL INVALID_LOGON;
      ELSE
         CALL sp_login_environment;
      END IF;
   END
go
GRANT EXECUTE ON DBA.login_check TO PUBLIC
go
SET OPTION PUBLIC.login_procedure='DBA.login_check'
go
```

1.1.5.2.8 Network Aspects of Integrated Logins

If the database is located on a network server, then one of two conditions must be met for integrated logins to be used.

- The user profile used for the integrated login connection attempt must be identical on both the local computer and the server. The passwords for both user profiles must also be identical.
  For example, when the user DSmith attempts to connect using an integrated login to a database loaded on a network server, identical user profile names and passwords must exist on both the local computer and the computer running the database server. The user DSmith must be permitted to log in to both computers.

- If network access is controlled by a Microsoft Domain Controller, the user attempting an integrated login must have domain permissions with the Domain Controller server and be logged in to the network. A user profile on the network server matching the user profile on the local computer is not required.

1.1.5.3 LDAP User Authentication

Lightweight Directory Access Protocol (LDAP) is an industry standard for accessing directory services.

LDAP user authentication allows client applications to send user ID and password information to the database server for authentication by the LDAP server instead of using the catalog. Authentication using the LDAP server allows organization wide password management.

LDAP user authentication is ideal for organizations with an existing computing environment who want to simplify and centralize user administration, or for users in a new computing environment who want to avoid unnecessary complexities for administering users in disparate applications and databases.

The login_mode database option includes an LDAPUA mode, which extends the possible authentication mechanisms to include LDAP User Authentication. The SQL Anywhere LDAP user authentication support allows SQL Anywhere to be integrated into existing enterprise-wide directory access frameworks based on LDAP.

How LDAP User Authentication Works

Each user is associated with a login policy. The login policy can optionally reference a primary LDAP server object, a secondary LDAP server object, or both for user authentication. You can define multiple login policies with different options to specify multiple domains.

To authenticate a user whose login policy specifies an LDAP server object, the **Distinguished Name (DN)** for that user and the password associated with the DN are used to connect to a trusted LDAP directory server.

The client application typically does not know or send its LDAP DN to the database server; instead, it knows and sends its database user ID and password. The association of the database user ID with its DN is then done by searching for the DN of a given user ID on the LDAP server using a predefined search string. The DN returned to the database server from a search on the LDAP server is then stored in the ISYSUSER system table (user_dn column) along with the current UTC time (user_dn_cached_at column). The stored DN value continues to be used until the associated login policy is updated using an ALTER LOGIN POLICY...LDAP_REFRESH_DN=NOW statement.

LDAP Server Configuration Objects

The configuration information for connecting to an LDAP server is stored in an **LDAP server configuration object**. Each row in the ISYSLDAPSERVER system table reflects the settings for an LDAP server configuration object. Each configuration object contains a set of attributes for connecting, including the LDAP search string used to obtain the DN for a given user ID. These attributes and the state of an LDAP server can be examined by querying the SYSLDAPSERVER view.

High Availability and LDAP User Authentication

High availability for an LDAP server is provided by designating it as either a primary or secondary server for a given user via the user's login policy. In the event that a primary LDAP server is unreachable for any reason (for example, LDAP server maintenance, a network failure, or an LDAP server crash), the secondary LDAP server is used for user authentication. The ability to fail over from a primary to a secondary and then fail back from a secondary to a primary can be managed manually using SQL statements or can be performed automatically by the database server when it detects that a change is appropriate.

Inheritance on LDAP Login Policy Options

Inheritance from the default root login policy occurs for LDAP policy options as a group. For example, if the root login policy defines a primary LDAP server and a login policy, policy_X, is defined that does not specify LDAP policy options, then users associated with policy_X inherit the primary LDAP server and LDAP policy options from the root policy.

If another login policy, policy_Y, defines only a secondary LDAP server, then users associated with policy_Y only use the secondary LDAP server and related settings from that policy; none of the LDAPUA policy options from the root policy are inherited.

In this section:

1.1.5.3.1 Activating or Suspending an LDAP Server Configuration Object (SQL Central)

Activate or suspend an LDAP server configuration object for use with an LDAP server.

Prerequisites

You must have the MANAGE ANY LDAP SERVER system privilege.

Context

You must re-enable an LDAP server configuration object after modifying it, or suspend an LDAP server configuration object to modify it or drop it.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *LDAP Servers*.
3. In the right pane, right-click the LDAP server configuration, and then click *Activate* or *Suspend* accordingly.
4. Click *OK*.

Results

The status of the LDAP server configuration object is updated.

Related Information

ALTER LDAP SERVER Statement
SYSLDAPSERVER System View

1.1.5.3.2 Altering an LDAP Server Configuration Object (SQL Central)

Alter an LDAP server configuration object to connect to an LDAP server for LDAP user authentication.

Prerequisites

You must have the MANAGE ANY LDAP SERVER system privilege.

Context

SQL Central suspends an LDAP server configuration object before altering it and reactivates it once the operation completes.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *LDAP Servers*.
3. In the right pane, right-click the LDAP server configuration you want to alter, and then click *Properties*.
4. Edit the properties on the *General* and *Settings* tabs as required, and then click *OK*.

Results

The settings for the LDAP server configuration object are updated in the ISYSLDAPSERVER system table.

Related Information

Activating or Suspending an LDAP Server Configuration Object (SQL Central) [page 140]
ALTER LDAP SERVER Statement
SYSLDAPSERVER System View

1.1.5.3.3 Validating an LDAP Server Configuration Object (SQL Central)

Connect to an LDAP server to ensure that the LDAP configuration object is configured properly.

Prerequisites

You must have the MANAGE ANY LDAP SERVER system privilege.

The LDAP server associated with the LDAP server configuration object must be accessible.

Context

The database server connects to the LDAP server using the settings in the LDAP configuration object to ensure that they are valid.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *LDAP Servers*.
3. In the right pane, right-click the LDAP server configuration, and then click *Test Connection*.

Results

The database server validates the LDAP server configuration object. If problems occur during the validation, the database server returns warnings.

Related Information

VALIDATE LDAP SERVER Statement
SYSLDAPSERVER System View

1.1.5.3.4 Dropping an LDAP Server Configuration Object (SQL Central)

Drop an LDAP server configuration object used to connect to an LDAP server.

Prerequisites

You must have the MANAGE ANY LDAP SERVER system privilege.

Context

SQL Central removes all references to the LDAP server configuration object from any login policies before dropping it.

SQL Central automatically suspends the LDAP server configuration object before dropping it.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *LDAP Servers*.
3. In the right pane, right-click the LDAP server configuration, and then click *Delete*.
4. Confirm your choice by clicking *Yes*.

Results

The database server drops the LDAP server configuration object, and references to it are removed from the ISYSLDAPSERVER system table.

Related Information

DROP LDAP SERVER Statement
SYSLDAPSERVER System View

1.1.5.4 Tutorial: Creating an LDAP User Authentication Environment

This tutorial guides you through the basic steps for setting up an LDAP user authentication environment in Interactive SQL or SQL Central.

In this section:

1.1.5.4.1 Creating an Authentication Environment Using Interactive SQL

Use Interactive SQL to create an LDAP User Authentication Environment

Prerequisites

To accomplish these tasks, you must have the following system privileges:

- SET ANY SECURITY OPTION
- MANAGE ANY USER
- MANAGE ANY LOGIN POLICY
- MANAGE ANY LDAP SERVER

Context

This tutorial shows you how to:

- Create an LDAP server configuration object.
- Create a login policy that uses the LDAP server.
- Create users that authenticate to the LDAP server by using the login policy that you defined.

1. Lesson 1: Creating an LDAP Server Configuration Object Using Interactive SQL [page 146]
   Use Interactive SQL to create a server configuration object
2. Lesson 2: Creating a Login Policy That Uses an LDAP Server Using Interactive SQL [page 148]
   Use Interactive SQL to create a login policy that uses the LDAP server.
3. Lesson 3: Creating a User That Authenticates to an LDAP Server Using Interactive SQL [page 149]
   Use Interactive SQL to create users that authenticate to the LDAP server by using that login policy.

Related Information

Lesson 1: Creating an LDAP Server Configuration Object Using Interactive SQL [page 146]

1.1.5.4.1.1 Lesson 1: Creating an LDAP Server Configuration Object Using Interactive SQL

Use Interactive SQL to create a server configuration object

Prerequisites

You must have an LDAP server.

You must have completed the previous lessons in this tutorial.

You must have the system privileges listed at the beginning of this tutorial.

Procedure

1. (Optional) To enable TLS encryption for communications with your LDAP server, specify a certificate file for the server to use. For example:

   ```
   SET OPTION PUBLIC.trusted_certificates_file='c:\\certificates\\trusted.txt';
   ```

   To enable TLS encryption in the steps that follow, change the TLS OFF clause to TLS ON in each of the examples where this clause occurs.
2. Execute a VALIDATE LDAP SERVER statement to test the connection to an LDAP server.

   For example, the following statement verifies the connection attributes of an existing LDAP server. The database server connects to the LDAP server with the supplied credentials.

   ```
   VALIDATE LDAP SERVER
       SEARCH DN
           URL 'ldap://iq10web:389/dc=sap,dc=com?dn?sub?uid=*'
           ACCESS ACCOUNT 'cn=Manager,dc=sap,dc=com'
           IDENTIFIED BY 'Not4YourEyes'
       AUTHENTICATION URL 'ldap://iq10web:389/'
       CONNECTION TIMEOUT 1000
       CONNECTION RETRIES 3
       TLS OFF;
   ```

   This LDAP server uses port 389 for communications. The example does not enable TLS encryption (TLS OFF). Your VALIDATE LDAP SERVER statement must execute without error before continuing to the next step.
3. Execute a CREATE LDAP SERVER statement to create an LDAP server configuration object.

   For example, the following statement defines an LDAP server configuration object that can be used for user authentication.

   ```
   CREATE LDAP SERVER prim_ldap
       SEARCH DN
           URL 'ldap://iq10web:389/dc=sap,dc=com?dn?sub?uid=*'
           ACCESS ACCOUNT 'cn=Manager,dc=sap,dc=com'
           IDENTIFIED BY 'Not4YourEyes'
   ```

```
    AUTHENTICATION URL 'ldap://iq10web:389/'
    CONNECTION TIMEOUT 1000
    CONNECTION RETRIES 3
    TLS OFF;
```

Unlike the VALIDATE LDAP SERVER statement, the CREATE LDAP SERVER statement does not attempt a connection to the LDAP server.

4. Execute a CREATE LDAP SERVER statement to create a second LDAP server configuration object to be used for failover. This step is optional but is required for the steps that follow.

   For example, the following statement defines an LDAP server configuration object that can be used as a failover for user authentication.

```
CREATE LDAP SERVER sec_ldap
    SEARCH DN
        URL 'ldap://iq10web:390/dc=sap,dc=com?dn?sub?uid=*'
        ACCESS ACCOUNT 'cn=Manager,dc=sap,dc=com'
        IDENTIFIED BY 'Not4YourEyes'
    AUTHENTICATION URL 'ldap://iq10web:390/'
    CONNECTION TIMEOUT 1000
    CONNECTION RETRIES 3
    TLS OFF;
```

   This LDAP server uses port 390 for communications.

5. Execute a SET OPTION statement to change the login mode to enable LDAP user authentication.

   For example, the following statement enables both standard and LDAP user authentication.

```
SET OPTION PUBLIC.login_mode='Standard,LDAPUA';
```

   For additional security, use SET TEMPORARY OPTION to set the login_mode option. If the database is copied, then this setting prevents unauthorized access from a spoofed LDAP server.

```
SET TEMPORARY OPTION PUBLIC.login_mode='Standard,LDAPUA';
```

   The current setting of the login_mode option can be queried as follows:

```
SELECT connection_property('login_mode');
```

Results

An LDAP server configuration object is created, and references to it are added to the ISYSLDAPSERVER system table.

Next Steps

Proceed to the next lesson.

Task overview: Creating an Authentication Environment Using Interactive SQL [page 145]

Next task: Lesson 2: Creating a Login Policy That Uses an LDAP Server Using Interactive SQL [page 148]

Related Information

VALIDATE LDAP SERVER Statement
CREATE LDAP SERVER Statement
login_mode Option [page 752]
trusted_certificates_file Option [page 856]
CREATE LOGIN POLICY Statement
CREATE USER Statement
sa_get_ldapserver_status System Procedure

1.1.5.4.1.2 Lesson 2: Creating a Login Policy That Uses an LDAP Server Using Interactive SQL

Use Interactive SQL to create a login policy that uses the LDAP server.

Procedure

Execute a CREATE LOGIN POLICY statement to create a new login policy.

For example, the following statement creates a new login policy that can be used to authenticate users.

```
CREATE LOGIN POLICY ldap_policy_both
  LDAP_PRIMARY_SERVER=prim_ldap
  LDAP_SECONDARY_SERVER=sec_ldap
  LDAP_FAILOVER_TO_STD=ON;
```

The names of the primary and secondary LDAP servers are specified in this login policy. If authentication for the user associated with this login policy fails, then standard authentication is attempted (if it is permitted by the login mode). The current settings of the login policy can be queried as follows:

```
SELECT lpo.* FROM SYS.SYSLOGINPOLICYOPTION AS lpo,
    SYS.SYSLOGINPOLICY AS lp
  WHERE lpo.login_policy_id = lp.login_policy_id
    AND lp.login_policy_name = 'ldap_policy_both';
```

Results

A login policy is created that uses an LDAP server for user authentication and definitions for it are added to the SYSLOGINPOLICY and SYSLOGINPOLICYOPTION system tables.

Next Steps

Proceed to the next lesson.

Task overview: Creating an Authentication Environment Using Interactive SQL [page 145]

Previous task: Lesson 1: Creating an LDAP Server Configuration Object Using Interactive SQL [page 146]

Next task: Lesson 3: Creating a User That Authenticates to an LDAP Server Using Interactive SQL [page 149]

Related Information

CREATE LOGIN POLICY Statement
SYSLOGINPOLICY System View
SYSLOGINPOLICYOPTION System View

1.1.5.4.1.3 Lesson 3: Creating a User That Authenticates to an LDAP Server Using Interactive SQL

Use Interactive SQL to create users that authenticate to the LDAP server by using that login policy.

Procedure

1. Execute a CREATE USER statement to create a new user ID with the LDAP login policy defined in a previous step.

   For example, the following statement creates a new user ID that authenticates against either the primary or secondary LDAP server.

   ```
   CREATE USER ldap_user01 LOGIN POLICY ldap_policy_both;
   ```

   The IDENTIFIED BY clause is omitted. The IDENTIFIED BY clause then specifies the password to use for standard authentication. If a password is specified using this clause, it need not be the same as the password authenticated by the LDAP server. Since this password is replaced the first time the user successfully authenticates to the LDAP server, the IDENTIFIED BY clause is omitted here.
2. Activate the LDAP servers for immediate use. The following statements activate the primary and secondary LDAP servers.

   ```
   ALTER LDAP SERVER prim_ldap WITH ACTIVATE;
   ALTER LDAP SERVER sec_ldap WITH ACTIVATE;
   ```

The following query determines the current state of all LDAP servers.

```
CALL sa_get_ldapserver_status;
```

The ldsrv_state column of the result set indicates that the two LDAP servers are in the READY state.

3. Execute an Interactive SQL CONNECT statement to connect to the database with LDAP user authentication.

   For example, the following statement connects to the sample database by using the specified user ID and password.

   ```
   CONNECT DATABASE demo USER ldap_user01 IDENTIFIED BY 'abcd1234';
   ```

   If the LDAP server fails the user authentication, then standard authentication is attempted. Standard authentication can fail if the standard password does not match the one provided (for example, because it has not been updated during LDAP authentication).

Results

A database user is created that uses an LDAP server to authenticate. Whenever the LDAP server is not available, standard authentication takes place.

Task overview: Creating an Authentication Environment Using Interactive SQL [page 145]

Previous task: Lesson 2: Creating a Login Policy That Uses an LDAP Server Using Interactive SQL [page 148]

Related Information

CREATE USER Statement
SYSUSER System View

1.1.5.4.2 Creating an Authentication Environment Using SQL Central

Use SQL Central to create an LDAP User Authentication Environment

Prerequisites

To accomplish these tasks, you must have the following system privileges:

- SET ANY SECURITY OPTION
- MANAGE ANY USER
- MANAGE ANY LOGIN POLICY
- MANAGE ANY LDAP SERVER

Context

This tutorial shows you how to:

- Create an LDAP server configuration object.
- Create a login policy that uses the LDAP server.
- Create users that authenticate to the LDAP server by using the login policy that you defined.

1. Lesson 1: Creating an LDAP Server Configuration Object Using SQL Central [page 152]
   Use SQL Central to create two LDAP server configuration objects.
2. Lesson 2: Creating a Login Policy That Uses an LDAP Server Using SQL Central [page 153]
   Use SQL Central to create a login policy that uses the LDAP server.
3. Lesson 3: Creating a User That Authenticates to an LDAP Server Using SQL Central [page 155]
   Use SQL Central to create a user that authenticates to an LDAP server.

Related Information

Lesson 1: Creating an LDAP Server Configuration Object Using SQL Central [page 152]

1.1.5.4.2.1 Lesson 1: Creating an LDAP Server Configuration Object Using SQL Central

Use SQL Central to create two LDAP server configuration objects.

Prerequisites

You must have the system privileges listed at the beginning of this tutorial.

Context

Each LDAP server is accessed via TCP/IP. An LDAP server definition is required for each LDAP server that you want to use for user authentication. The responses shown below are for demonstration purposes only. Supply your own appropriate equivalent responses.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, right-click *LDAP Servers*, and then click *New* > *LDAP Server*.
3. For the name of the LDAP server, type **`prim_ldap`** and then click *Next*.
4. Select the type of network encryption: *No encryption (ldap:)*.
5. For the host name and port number of the LDAP server, type the server name and port: **`iq10web`** and **`389`**.
6. For *Authentication URL*, a URL appears (for example, ldap://iq10web:389/). Click *Next*.
7. Type the search URL directly into the *Search URL* field: **`ldap://iq10web:389/dc=sap,dc=com?dn?sub?uid=*`**.
8. For *Distinguished Name*, type the distinguished name: **`cn=Manager,dc=sap,dc=com`**.
9. If a password is required to authenticate to the LDAP server, then type it in the *Password* field (for example, **`Not4YourEyes`**).
10. Click *Next*.
11. For the connection timeout, select the connection timeout interval.
12. For the number of connection retries, select the number of times, and then click *Next*.
13. To activate the LDAP server, select *Activate this LDAP server now*.
14. Select the *Login modes* (at least *Standard* and *LDAPUA* should be selected).
15. Click *Test Connection* to verify your LDAP connection parameters. If you have entered your connection parameters correctly, then the connection succeeds. Click *Close*, then click *Next*.
16. (Optional) Add a comment. Click *Next*.
17. The SQL that is executed to create the LDAP server configuration object appears. Click *Finish*.

Results

An LDAP server configuration object is created, and references to it are added to the ISYSLDAPSERVER system table.

Next Steps

Proceed to the next lesson.

Task overview: Creating an Authentication Environment Using SQL Central [page 151]

Next task: Lesson 2: Creating a Login Policy That Uses an LDAP Server Using SQL Central [page 153]

Related Information

VALIDATE LDAP SERVER Statement
CREATE LDAP SERVER Statement
login_mode Option [page 752]
trusted_certificates_file Option [page 856]
CREATE LOGIN POLICY Statement
CREATE USER Statement
sa_get_ldapserver_status System Procedure

1.1.5.4.2.2 Lesson 2: Creating a Login Policy That Uses an LDAP Server Using SQL Central

Use SQL Central to create a login policy that uses the LDAP server.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the system privileges listed at the beginning of this tutorial.

Context

A login policy is required when you use an LDAP server for user authentication. The responses shown below are for demonstration purposes only. Supply your own appropriate equivalent responses.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, right-click *Login policies* and click *New* > *Login Policy*.
3. Enter the name of the login policy (`ldap_policy_both` for example), and then click *Next*.
4. Select the *LDAP primary server* (`prim_ldap` for example).
5. (Optional) Select the *LDAP secondary server* (`sec_ldap` for example).
6. Click *Next*.
7. (Optional) Add a comment. Click *Next*.
8. The SQL that is executed to create the login policy appears. Click *Finish*.

Results

A login policy is created that uses an LDAP server for user authentication and definitions for it are added to the SYSLOGINPOLICY and SYSLOGINPOLICYOPTION system tables.

Next Steps

Proceed to the next lesson.

Task overview: Creating an Authentication Environment Using SQL Central [page 151]

Previous task: Lesson 1: Creating an LDAP Server Configuration Object Using SQL Central [page 152]

Next task: Lesson 3: Creating a User That Authenticates to an LDAP Server Using SQL Central [page 155]

Related Information

CREATE LOGIN POLICY Statement
SYSLOGINPOLICY System View
SYSLOGINPOLICYOPTION System View

1.1.5.4.2.3 Lesson 3: Creating a User That Authenticates to an LDAP Server Using SQL Central

Use SQL Central to create a user that authenticates to an LDAP server.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the system privileges listed at the beginning of this tutorial.

Context

Users that authenticate to an LDAP server must be identified to the database. The responses shown below are for demonstration purposes only. Supply your own appropriate equivalent responses.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, right-click *Users* and click *New* > *User*.
3. Enter the name of the new user (`ldap_user01` for example), and then click *Next*.
4. Ensure that *Assign a password to this user* is not selected.
5. For *Login policy*, select the login policy (`ldap_policy_both` for example).
6. Click *Next*.
7. (Optional) Add a comment. Click *Next*.
8. The SQL that is executed to create the user appears. Click *Finish*.

Results

A database user is created that uses an LDAP server to authenticate. Whenever the LDAP server is not available, standard authentication takes place.

Task overview: Creating an Authentication Environment Using SQL Central [page 151]

Previous task: Lesson 2: Creating a Login Policy That Uses an LDAP Server Using SQL Central [page 153]

Related Information

CREATE USER Statement
SYSUSER System View

1.1.5.5 PAM User Authentication

Pluggable Authentication Module (PAM) is an authentication mechanism that provides a common interface for multiple low-level authentication schemes, allowing a user to use the same credentials on all of their systems.

PAM authentication centralizes user authentication in a separate external system-wide module. Any authentication mechanism can be plugged into a PAM system by writing an authentication module and configuring PAM to use that module.

PAM user authentication (PAMUA) is available on all supported Unix and Linux platforms. To configure PAMUA for your platform, see your operating system documentation.

In this section:

Related Information

Login Policy Options and Default Values [page 642]
login_mode Option [page 752]

1.1.5.5.1 Enabling PAM User Authentication (SQL)

Configure your database to use PAM user authentication.

Prerequisites

- PAM must be configured on the Unix system.
- Step 2 requires the SET ANY SECURITY OPTION system privilege.
- Steps 3 and 4 require the MANAGE ANY LOGIN POLICY system privilege.

Context

For the following example, assume that you have configured the PAM service name **pam_rule**.

Procedure

1. In Interactive SQL, connect to your database.
2. Create a login policy that assigns the **pam_rule** service name to the pam_service_name login option and determines whether or not the database fails over to standard authentication if PAMUA cannot perform authentication. For example:

   ```
   CREATE LOGIN POLICY pam_policy
   PAM_SERVICE_NAME = pam_rule
   PAM_FAILOVER_TO_STD = ON;
   ```

3. Create a new user and assign the new login policy to them.

   ```
   CREATE USER pam_userID LOGIN POLICY pam_policy;
   ```

4. Add the PAMUA value to the login_mode database option.

   ```
   SET TEMPORARY OPTION public.login_mode = "PAMUA,STANDARD";
   ```

5. Verify that the user can log on with PAMUA by using the dbping utility to test the database connection.

   ```
   dbping -c "UID=pam_userID;PWD=pam-user-password;LINKS=tcpip(host=computer-
   name;PORT=port-number);DBN=database-name -d
   ```

Results

The user pam_userID can now log in to the database using their Unix credentials with PAMUA.

Related Information

CREATE LOGIN POLICY Statement
Login Policy Options and Default Values [page 642]
login_mode Option [page 752]

1.1.5.5.2 Configuring PAM User Authentication (SQL Central)

Configure your database to use PAM user authentication.

Prerequisites

- PAM must be configured on the Unix system including a PAM service
- Step 2 requires the SET ANY SECURITY OPTION system privilege.
- Steps 3 and 4 require the MANAGE ANY LOGIN POLICY system privilege.

Procedure

1. Connect to the database.
2. Create a login policy for PAM.
   a. In the left pane, right-click *Login Policies* > *New* > *Login Policy*
   b. Specify a name for login policy, such as pam_login_policy
   c. When prompted to set the options for the login policy, specify your PAM service name for the *PAM service name*.
3. Assign the login policy to a user.
   a. In the left pane, click *Users*
   b. In the right pane, right-click the user and click *Properties*.
   c. Select the PAM login policy from the *Login policy* dropdown list.
4. Add the PAMUA value to the login_mode database option.
   a. In the left pane right-click the database, and click *Options*.
   b. Select the *login_mode* option and add *PAMUA* to the list of values.
   c. Click *Set Permanent Now*.
5. Verify that the user can log on with PAMUA by using the dbping utility to test the database connection.

```
dbping -c "UID=pam_userID;PWD=pam-user-password;LINKS=tcpip(host=computer-
name;PORT=port-number);DBN=database-name -d
```

Results

The user pam_userID can now log in to the database using their Unix credentials with PAMUA.

Related Information

CREATE LOGIN POLICY Statement
Login Policy Options and Default Values [page 642]
login_mode Option [page 752]

1.1.5.6 Kerberos User Authentication

The Kerberos login feature allows you to maintain a single user ID and password for database connections, operating system, and network logins.

The Kerberos login is more convenient for users and permits a single security system for database and network security. Its advantages include:

- The user does not need to provide a user ID or password to connect to the database.
- Multiple users can be mapped to a single database user ID.
- The name and password used to log in to Kerberos do not have to match the database user ID and password.

Kerberos is a network authentication protocol that provides strong authentication and encryption using secret-key cryptography. Users already logged in to Kerberos can connect to a database without providing a user ID or password.

Kerberos can be used for authentication. To delegate authentication to Kerberos you must:

- configure the server and database to use Kerberos logins.
- create mapping between the user ID that logs in to the computer or network, and the database user.

> ⚠ Caution
>
> When using Kerberos logins as a single security solution, be sure to inform yourself on the security concern related to copied databases.

SQL Anywhere does not include the Kerberos software; it must be obtained separately. The following components are included with the Kerberos software:

Kerberos libraries

These are referred to as the Kerberos Client or GSS (Generic Security Services)-API runtime library. These Kerberos libraries implement the well-defined GSS-API. The libraries are required on each client and server computer that intends to use Kerberos. The built-in Windows SSPI interface can be used instead of a third-party Kerberos client library if you are using Active Directory as your KDC.

SSPI can only be used by clients in the Kerberos connection parameter. The database server cannot use SSPI. It needs a supported Kerberos client other than SSPI.

A Kerberos Key Distribution Center (KDC) server

The KDC functions as a storehouse for users and servers. It also verifies the identification of users and servers. The KDC is typically installed on a server computer not intended for applications or user logins.

Kerberos authentication from DBLib, ODBC, OLE DB, and ADO.NET clients, and SAP Open Client and jConnect clients is supported. Kerberos authentication can be used with SQL Anywhere transport layer security encryption, but Kerberos encryption for network communications is not supported.

Windows uses Kerberos for Windows domains and domain accounts. Active Directory Windows Domain Controllers implement a Kerberos KDC. A third-party Kerberos client or runtime is still required on the database server computer for authentication in this environment, but the Windows client computers can use the built-in Windows SSPI interface instead of a third-party Kerberos client or runtime.

In this section:

Related Information

Security: Use Login Modes to Secure the Database [page 171]

1.1.5.6.1 Kerberos Clients

Kerberos authentication is available on several platforms.

The following table lists the default names and locations of the keytab and GSS-API files used by the supported Kerberos clients.

> **i** Note
>
> SSPI can only be used by clients in the Kerberos connection parameter. The database server cannot use SSPI. It needs a supported Kerberos client other than SSPI.

Kerberos client	Default keytab file	GSS-API library file name	Notes
Windows MIT Kerberos client	`C:\WINDOWS\krb5kt`	`gssapi32.dll` or `gssapi64.dll`	The KRB5_KTNAME environment variable can be set before starting the database server to specify a different keytab file.
Windows CyberSafe Kerberos client	`C:\Program Files\CyberSafe\v5srvtab`	`gssapi32.dll` or `gssapi64.dll`	The CSFC5KTNAME environment variable can be set before starting the database server to specify a different keytab file.
UNIX/Linux MIT Kerberos client	`/etc/krb5.keytab`	`libgssapi_krb5.so`[1]	The KRB5_KTNAME environment variable can be set before starting the database server to specify a different keytab file.
UNIX/Linux CyberSafe Kerberos client	`/krb5/v5srvtab`	`libgss.so`[1]	The CSFC5KTNAME environment variable can be set before starting the database server to specify a different keytab file.
UNIX/Linux Heimdal Kerberos client	`/etc/krb5.keytab`	`libgssapi.so.1`[1]	

[1] These file names may vary depending on your operating system and Kerberos client version.

Related Information

SAP SQL Anywhere Supported Kerberos Clients

1.1.5.6.2 Setting up a Kerberos System

Configure Kerberos authentication to be used with SQL Anywhere.

Prerequisites

You must be logged in to your computer using Kerberos authentication.

Context

Kerberos is a network authentication protocol that provides strong authentication and encryption using secret-key cryptography.

Procedure

1. If necessary, install and configure the Kerberos client software, including the GSS-API runtime library, on both the client and server.

   On Windows client computers using an Active Directory Key Distribution Center (KDC), SSPI can be used and you do not need to install the Kerberos client.
2. If necessary, create a Kerberos principal in the Kerberos KDC for each user.

   A Kerberos principal is a Kerberos user ID in the format `user/instance@REALM`, where `/instance` is optional. If you are already using Kerberos, the principal should already exist, so you do not need to create a Kerberos principal for each user.

   Principals are case sensitive and must be specified in the correct case. Mappings for multiple principals that differ only in case are not supported (for example, you cannot have mappings for both jjordan@MYREALM.COM and JJordan@MYREALM.COM).
3. Create a Kerberos principal in the KDC for the SQL Anywhere database server.

   The default Kerberos principal for the database server has the format `server-name@REALM`, where `server-name` is the SQL Anywhere database server name. To use a different server principal, use the -kp server option. Principals are case sensitive, and `server-name` cannot contain multibyte characters, or the characters /, \, or @.

   You must create a server service principal within the KDC because servers use a keytab file for KDC authentication. The keytab file is protected and encrypted.
4. Securely extract and copy the keytab for the server principal (`server-name@REALM` by default, or the principal used with -kp server option) from the KDC to the computer running the SQL Anywhere database server. The default location of the keytab file depends on the Kerberos client and the platform. The keytab file's permissions should be set so that the SQL Anywhere server can read it, but unauthorized users do not have read permission.

Results

The Kerberos system is authenticated and configured to be used with SQL Anywhere.

Next Steps

Configure your SQL Anywhere database server and database to use Kerberos.

Related Information

Configuring SQL Anywhere Databases to Use Kerberos (SQL) [page 163]
Connecting Using SSPI for Kerberos Logins on Microsoft Windows [page 168]

1.1.5.6.3 Configuring SQL Anywhere Databases to Use Kerberos (SQL)

Configure databases to use Kerberos logins.

Prerequisites

You must have the SET ANY PUBLIC OPTION and MANAGE ANY USER system privileges.

You must already have Kerberos configured before SQL Anywhere can use it.

Context

The Kerberos login feature allows you to maintain a single user ID and password for database connections, operating systems, and network logins.

Procedure

1. Start the database server with the -krb or -kr option to enable Kerberos authentication, or use the -kl option to specify the location of the GSS-API library and enable Kerberos.
2. Change the public or temporary public option login_mode to a value that includes Kerberos. As database options apply only to the database in which they are found, different databases can have a different Kerberos login setting, even if they are loaded and running on the same database server. For example:

```
SET TEMPORARY OPTION PUBLIC.login_mode = 'Kerberos,Standard';
```

> ⚠ Caution
>
> Setting the login_mode database option to Kerberos restricts connections to only those users who have been granted a Kerberos login mapping. Attempting to connect using a user ID and password generates an error unless you are a user with SYS_AUTH_DBA_ROLE compatibility role.

3. Create a database user ID for the client user. You can use an existing database user ID for the Kerberos login, as long as that user has the correct privileges. For example:

```
CREATE USER "kerberos-user"
IDENTIFIED BY abc123;
```

4. Execute a GRANT KERBEROS LOGIN TO statement to create a mapping from the client's Kerberos principal to an existing database user ID. For example:

```
GRANT KERBEROS LOGIN TO "pchin@MYREALM.COM"
AS USER "kerberos-user";
```

   To connect when a Kerberos principal is used that does not have a mapping, ensure the Guest database user ID exists and has a password.

5. Ensure the client user has already logged on (has a valid Kerberos ticket-granting ticket) using their Kerberos principal and that the client's Kerberos ticket has not expired. A Windows user logged in to a domain account already has a ticket-granting ticket, which allows them to authenticate to servers, providing their principal has enough permissions.

   A ticket-granting ticket is a Kerberos ticket encrypted with the user's password that is used by the Ticket Granting Service to verify the user's identity.

6. Connect from the client, specifying the KERBEROS connection parameter (Often KERBEROS=YES, but KERBEROS=SSPI or KERBEROS=`GSS-API-library-file` can also be used). If the user ID or password connection parameters are specified, they are ignored. For example:

```
dbisql -c "KERBEROS=YES;Server=my_server_princ"
```

Results

The database is configured to use Kerberos authentication.

Example

A connection attempt using the following SQL statement is successful if the user logs in with a user profile name that matches a Kerberos login mapping for the default database on a database server:

```
CONNECT USING 'KERBEROS=YES';
```

The CONNECT statement can connect to a database if all the following conditions are true:

- A database server is currently running.
- The default database on the current database server is enabled to accept Kerberos authenticated connections.
- A Kerberos login mapping has been created for the user's current Kerberos principal.
- If the user is prompted with a window by the database server for more connection information (such as occurs when using Interactive SQL), the user clicks *OK* without providing more information.

Next Steps

You can use Kerberos authentication to connect from a client. Optionally, you can create a Kerberos login mapping.

Related Information

Creating a Kerberos Login Mapping (SQL Central) [page 166]
Setting up a Kerberos System [page 161]
Connecting Using SSPI for Kerberos Logins on Microsoft Windows [page 168]
-kl Database Server Option [page 457]
-kr Database Server Option (Deprecated) [page 459]
-krb Database Server Option [page 461]
login_mode Option [page 752]
GRANT CONNECT Statement

1.1.5.6.4 Connections from an SAP Open Client or jConnect Application

The database server accepts connections from an SAP Open Client or jConnect application.

- Set up Kerberos user authentication.
- Configure SQL Anywhere to use Kerberos.
- Set up SAP Open Client or jConnect as you would for Kerberos user authentication with Adaptive Server Enterprise. The server name must be the SQL Anywhere server's name and is case sensitive. You cannot connect using an alternate server name from Open Client or jConnect.

Related Information

Setting up a Kerberos System [page 161]
Configuring SQL Anywhere Databases to Use Kerberos (SQL) [page 163]
-krb Database Server Option [page 461]
-kr Database Server Option (Deprecated) [page 459]
-kl Database Server Option [page 457]
login_mode Option [page 752]
GRANT CONNECT Statement
CREATE USER Statement
Kerberos (KRB) Connection Parameter [page 92]
Troubleshooting: Kerberos Connections [page 169]

1.1.5.6.5 Creating a Kerberos Login Mapping (SQL Central)

Use SQL Central to create a Kerberos login mapping, which configures an individual user for Kerberos authentication to the database server.

Prerequisites

You must already have Kerberos configured before SQL Anywhere can use it. You must already have your database server and database configured to use Kerberos.

The database user and the Kerberos principal must already exist.

You must have the MANAGE ANY USER system privilege and either the SELECT ANY TABLE system privilege or the SELECT privilege on the SYSLOGINMAP system view.

If you need to change the login mode, you must also have the SET ANY SECURITY OPTION system privilege.

Context

A Kerberos login mapping is the connection between the user ID that logs in to the computer or network and the database user.

Procedure

1. Use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, right-click *Login Mappings* and click *New* > *Login Mapping*.
3. Select *Kerberos login mapping* and click *Next*.
4. In the *Which Kerberos principal user will be connecting to the database* field, type the name of the Kerberos principal for whom the Kerberos login is to be created.
5. In the *Which database user do you want to associate with the Kerberos principal* list, select the database user ID this Kerberos principal maps to.
6. Follow the remaining instructions in the *Create Login Mapping Wizard*.

Results

A Kerberos login mapping is created.

Related Information

Setting up a Kerberos System [page 161]
Configuring SQL Anywhere Databases to Use Kerberos (SQL) [page 163]
GRANT CONNECT Statement

1.1.5.6.6 Revoking a Kerberos Login Mapping (SQL Central)

Use SQL Central to revoke a Kerberos login mapping.

Prerequisites

You must already have Kerberos configured before SQL Anywhere can use it. You must already have your database server and database configured to use Kerberos.

The database user and the Kerberos principal must already exist.

You must have the MANAGE ANY USER system privilege and either the SELECT ANY TABLE system privilege or the SELECT privilege on the SYSLOGINMAP system view.

Context

A Kerberos login mapping is the connection between the user ID that logs in to the computer or network, and the database user.

Procedure

1. Use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *Login Mappings*.
3. In the right pane, right-click the login mapping and click *Delete*.
4. Click *Yes*.

Results

The Kerberos login mapping is revoked.

Related Information

Setting up a Kerberos System [page 161]
Configuring SQL Anywhere Databases to Use Kerberos (SQL) [page 163]
REVOKE Statement

1.1.5.6.7 Connecting Using SSPI for Kerberos Logins on Microsoft Windows

Connect using SSPI without a Kerberos client installed on the client computer.

Prerequisites

You must already have Kerberos configured before SQL Anywhere can use it. You must already have your database server and database configured to use Kerberos.

Context

In a Windows domain, SSPI can be used on Windows-based computers without a Kerberos client installed on the client computer. Windows domain accounts already have associated Kerberos principals.

SSPI can only be used by SQL Anywhere clients in the Kerberos connection parameter. SQL Anywhere database servers cannot use SSPI. They need a supported Kerberos client other than SSPI.

Procedure

Connect to the database from the client computer. For example:

```
dbisql -c "KERBEROS=SSPI;Server=my_server_princ"
```

When Kerberos=SSPI is specified in the connection string, a Kerberos login is attempted.

A connection attempt using the following SQL statement also succeeds, providing the user has logged on with a user profile name that matches a Kerberos login mapping for the default database on a database server:

```
CONNECT USING 'KERBEROS=SSPI';
```

Results

You can use SSPI for Kerberos authentication on Windows.

Related Information

Setting up a Kerberos System [page 161]
Configuring SQL Anywhere Databases to Use Kerberos (SQL) [page 163]

1.1.5.6.8 Troubleshooting: Kerberos Connections

If you get unexpected errors when attempting to enable or use Kerberos authentication, it is recommended that you enable additional diagnostic messages on the database server and client.

Specifying the -z option when you start the database server, or using `CALL sa_server_option( 'DebuggingInformation', 'ON' )` if the server is already running includes additional diagnostic messages in the database server message log. The LogFile connection parameter writes client diagnostic messages to the specified file.

As an alternative to using the LogFile connection parameter, you can run the Ping utility (dbping) with the -z parameter. The -z parameter displays diagnostic messages that should help identify the cause of the connection problem.

Difficulties Starting the Database Server

Symptom	Common solutions
"Unable to load Kerberos GSS-API library" message	• Ensure a Kerberos client is installed on the database server computer, including the GSS-API library. • The database server -z output lists the name of the library that it is attempting to load. Verify the library name is correct. If necessary, use the -kl option to specify the correct library name. • Ensure the directory and any supporting libraries is listed in the library path (%PATH% on Windows). • If the database server -z output states the GSS-API library was missing entry points, then the library is not a supported Kerberos Version 5 GSS-API library.

Symptom	Common solutions
"Unable to acquire Kerberos credentials for server name "server-name"" message	• Ensure there is a principal for server-name@REALM in the KDC. Principals are case sensitive, so ensure the database server name is in the same case as the user portion of the principal name. • Ensure the name of the SQL Anywhere server is the primary/user portion of the principal. • Ensure that the server's principal has been extracted to a keytab file and the keytab file is in the correct location for the Kerberos client. • If the default realm for the Kerberos client on the database server computer is different from the realm in the server principal, use the -kr option to specify the realm in the server principal.
"Kerberos login failed" client error	• Check the database server diagnostic messages. Some problems with the keytab file used by the server are not detected until a client attempts to authenticate.

Troubleshooting Kerberos Client Connections

If the client got an error attempting to connect using Kerberos authentication:

Symptom	Common solutions
"Kerberos logins are not supported" error and the LogFile includes the message "Failed to load the Kerberos GSS-API library"	• Ensure a Kerberos client is installed on the client computer, including the GSS-API library. • The file specified by LogFile lists the name of the library that it is attempting to load. Verify that the library name is correct, and use the Kerberos connection parameter to specify the correct library name, if necessary. • Ensure that the directory including any supporting libraries is listed in the library path (%PATH% on Windows). • If the LogFile output states the GSS-API library was missing entry points, then the library is not a supported Kerberos Version 5 GSS-API library.
"Kerberos logins are not supported" error	• Ensure the database server has enabled Kerberos logins by specifying one or more of the -krb, -kl, or -kr server options. • Ensure Kerberos logins are supported on both the client and server platforms.

Symptom	Common solutions
"Kerberos login failed" error	• Ensure the user is logged into Kerberos and has a valid ticket-granting ticket that has not expired. • Ensure the client computer and server computer both have their time synchronized to within less than 5 minutes.
"Login mode 'Kerberos' not permitted by login_mode setting" error	• The public or temporary public database option setting for the login_mode option must include the value Kerberos to allow Kerberos logins.
"The login ID '`client-Kerberos-principal`' has not been mapped to any database user ID"	• The Kerberos principal must be mapped to a database user ID using the GRANT KERBEROS LOGIN statement. Note the full client principal including the realm must be provided to the GRANT KERBEROS LOGIN statement, and principals which differ only in the instance or realm are treated as different. • Alternatively, if you want any valid Kerberos principal which has not be explicitly mapped to be able to connect, create the guest database user ID with a password using GRANT CONNECT.

Related Information

-z Database Server Option [page 532]
Kerberos Clients [page 160]
login_mode Option [page 752]
Ping Utility (dbping) [page 1195]

1.1.5.7 Security: Use Login Modes to Secure the Database

There are several measures you can take to secure your database.

Setting the value of the login_mode option for a given database to allow a combination of Standard, Integrated, Kerberos, LDAPUA, and PAMUA logins by using the SET OPTION statement permanently enables the specified types of logins for that database. When you enable Integrated, Kerberos, LDAPUA, or PAMUA logins for your database, you rely on the security model of the operating system or network. For example, the following statement permanently enables Standard and Integrated logins:

```
SET OPTION PUBLIC.login_mode = 'Standard,Integrated';
```

If the database is shut down and restarted, the option value remains the same and Integrated logins remain enabled.

Setting the login_mode option using SET TEMPORARY OPTION still allows user access via Integrated logins, but only until the database is shut down. The following statement changes the option value temporarily:

```
SET TEMPORARY OPTION PUBLIC.login_mode = 'Standard,Integrated';
```

If the permanent option value is Standard, the database will revert to that value when it is shut down.

Setting temporary public options can provide additional security for your database. If the database file is copied to another computer, then Integrated, Kerberos, LDAPUA, and PAMUA logins will not be enabled by default.

If a database contains sensitive information, the computer where the database files are stored should be protected from unauthorized access. Otherwise, the database files could be copied and unauthorized access to the data could be obtained on another computer.

To increase database security:

- Make passwords complex and difficult to guess.
- Strongly encrypt the database file using the AES encryption features of SQL Anywhere. The encryption key should be complex and difficult to guess.
- Set the permanent PUBLIC.login_mode database option to Standard. To enable Integrated or Kerberos logins, only the temporary public option should be changed each time the server is started. This ensures that only Standard logins are allowed if the database is copied.

1.1.5.8 Security: Integrated Logins Can Result in Unrestricted Database Access

The Integrated login feature uses native Windows authentication to control access to a database.

The standard SQL Anywhere user ID/password database security is bypassed. The credentials of the user logged in to Windows are used to authenticate to a Windows domain controller. If a mapping exists for the authenticated user to a database user ID, then the user is permitted access to the database.

When using Integrated login, database administrators should give special consideration to the way Windows enforces login security to prevent unwanted access to the database.

> ⚠ Caution
>
> Granting integrated login to a user name without specifying a domain is not recommended.

Suppose that Integrated login is enabled for a database and a mapping of a Windows user to a database user ID is created as follows.

```
SET TEMPORARY OPTION PUBLIC.login_mode = 'Standard,Integrated';
GRANT INTEGRATED LOGIN TO DSmith AS USER BROWSER;
```

If a user successfully logs in to a Windows computer as DSmith, regardless of the Windows domain, they can authenticate to the database running on the same network as their Windows computer without further proof of identification.

Now suppose that Integrated logins are enabled in the database as follows.

```
SET TEMPORARY OPTION PUBLIC.login_mode = 'Standard,Integrated';
```

```
GRANT INTEGRATED LOGIN TO "MyDomain\DSmith" AS USER BROWSER;
```

In order to authenticate to the database, the user must first successfully log in to a Windows computer as DSmith in the domain MyDomain (MyDomain\DSmith). The user can then connect to the database running on the same network as their Windows computer without further proof of identification. Inclusion of the domain name ensures that the correct and expected user credentials are specified.

> ⚠ Caution
>
> Creating the Guest user ID with a password permits Integrated login access to the database for anyone that can authenticate to a Windows domain.

```
SET TEMPORARY OPTION PUBLIC.login_mode = 'Standard,Integrated';
CREATE USER Guest IDENTIFIED BY 'secret';
```

Creating the Guest user ID with a password and including Integrated in the login_mode setting enables Integrated login to the database for anyone that can authenticate to a Windows domain on the network. When an Integrated login mapping does not exist for the Windows user name, the Guest user is used (if it exists). The Guest user does not require an Integrated login mapping (a GRANT INTEGRATED LOGIN statement is not required for Guest).

If the Guest user does not exist, then the Integrated login will fail if there is no mapping for the Windows user.

Although the Guest user is limited by the roles and privileges assigned to it, creating the Guest user ID is not recommended. For example, database system tables can be inspected.

1.1.6 Communication Protocols

Database servers communicate over a variety of protocols, and many options for customizing the behavior of supported protocols are included.

In this section:

Related Information

The Database Server as an HTTP Web Server
Access to Web Services Using Web Clients
Interface Library Communication Protocols

1.1.6.1 Communication Protocol Considerations

Communication between a client application and a database server requires a communications protocol.

Communications protocols for communications across networks and for same-computer communications are supported.

Available Protocols for the Network Database Server (dbsrv17)

By default, the network database server starts the shared memory and TCP/IP protocols. You can provide TCP/IP protocol options or disable the TCP/IP protocol using the -x option.

Shared memory

This protocol is for same-computer communications and it is always available on all platforms.

TCP/IP

This protocol is used primarily for communications between different computers. You must use TCP/IP if you are using TDS clients (SAP Open Client or jConnect JDBC driver). It is available on all platforms.

Available Protocols for the Personal Database Server (dbeng17)

By default, the personal database server allows only shared memory connections. To also start the TCP/IP protocol, use the -x option.

Shared memory

This protocol is for same-computer communications and it is always available on all platforms.

For same-computer communications, shared memory tends to provide better performance than TCP/IP.

TCP/IP

This protocol is available only for same-computer communications on the personal server. In addition, you must use TCP/IP if you are using TDS clients (SAP Open Client or jConnect JDBC driver). It is available on all platforms.

Connecting from a SQL Anywhere Client

Shared memory

Shared memory is the default connection protocol. To connect over shared memory, do not include either the Host or CommLinks (LINKS) connection parameters in the client's connection string.

TCP/IP

To connect over TCP/IP, specify either the Host or the CommLinks (LINKS) connection parameter. The Host connection parameter is recommended.

In this section:

Related Information

SAP Open Client Support
TCP/IP Protocol [page 176]
Network Protocol Options [page 187]
Alphabetical List of Connection Parameters [page 45]
-x Database Server Option [page 523]
Host Connection Parameter [page 86]
CommLinks (LINKS) Connection Parameter [page 59]

1.1.6.1.1 Session and Terminal Considerations for Shared Memory

There are several session and terminal considerations for shared memory.

Shared Memory, Services, and Terminal Services (Windows 7 or Later)

- Any client can connect over shared memory to any database server that runs as a service.
- Any client can connect over shared memory to a database server that is running in the same session as the client. When attempting a shared memory connection, a client looks first for a database server with the specified name in the same session as the client. If none is found, the client then looks for a service with the specified name.
- A client running in one terminal session cannot connect over shared memory to a database server running in another terminal session. Use TCP/IP instead.
- A client that runs as a service cannot connect over shared memory to a database server running in a terminal session. Use TCP/IP instead.
- A database server *cannot* start if there is already a database server running with the same name in the current session or if there is a service running with the same name.
- A database server *can* start if there is a database server running with the same name in another session, as long as there is not a service running with the same name, and the database servers do not both use TCP/IP.

Shared Memory, Services, and Terminal Services (Windows 2003)

When using database servers and clients running as services or in terminal services:

- A client can connect over shared memory to a database server that is running in the same session as the client or that is running in the primary session. When attempting a shared memory connection, a client looks first for a database server with the specified name in the same session as the client, and then in the primary session.
- A client running in the primary session cannot connect over shared memory to a database server running in a non-primary session. A client running in a non-primary session cannot connect over shared memory to a database server running in another non-primary session. Use TCP/IP instead.
- A database server *cannot* start if there is already a database server with the same name running in the current session or in the primary session. A database server running on the desktop (of the primary session) could prevent a service with the same name from starting. However, since services usually start when the system starts, this problem rarely occurs.
- A database server *can* start using shared memory even if there is a database server with the same name running over shared memory in another non-primary session as long as there is not a database server with the same name running in the primary session.
- A database server *cannot* start using TCP/IP if there is a database server with the same name running over TCP/IP in any other session (primary or non-primary).

1.1.6.2 TCP/IP Protocol

TCP/IP is used to connect clients to databases running on different computers.

When you use the TCP/IP protocol, you can secure client/server communications using transport layer security and RSA encryption technology.

UDP

UDP is a transport layer protocol that sits on top of IP. UDP might be used to do initial server name resolution and then TCP is used for connection and communication.

UDP packets sent by the database server in response to client broadcasts contain no sensitive information. The data contained in these packets is limited to:

- the database server name
- the port number
- the database server version
- the names of databases running on the database server

You can hide database names from UDP broadcast responses by using the -dh database option.

You can specify the -sb server option to disable the UDP listeners.

Using TCP/IP with Microsoft Windows

The TCP/IP implementation for database servers on Windows uses Winsock 2.2.

In this section:

Client/Server Communications Encryption over TCP/IP [page 178]
You can secure communications between client applications and the database server over TCP/IP using Transport Layer Security (TLS).

IPv6 Support [page 178]
On IPv6-enabled computers, the network database server listens by default on all IPv4 and IPv6 addresses.

Firewall Connections [page 179]
There are restrictions on connections when the client application is on the other side of a firewall from the database server.

Dial-up Network Connections (CommLinks Connection Parameter) [page 181]
To connect over a dial-up network connection, you must use the CommLinks connection parameter.

Connections Using LDAP as a Name Server [page 181]
You can specify a central LDAP server to keep track of all database servers in an enterprise if you are operating on a Windows or Unix platform.

Communication Compression Settings [page 185]
Enabling compression for one or all connections, and setting the minimum size at which packets are compressed, can improve performance in some circumstances.

Related Information

-dh Database Option [page 550]
-sb Database Server Option [page 490]
-x Database Server Option [page 523]
Host Connection Parameter [page 86]
Broadcast (BCAST) Protocol Option [page 192]
Host (IP) Protocol Option (Client Side Only) [page 206]
MyIP (ME) Protocol Option [page 229]

1.1.6.2.1 Client/Server Communications Encryption over TCP/IP

You can secure communications between client applications and the database server over TCP/IP using Transport Layer Security (TLS).

By default, communication packets are not encrypted, which poses a potential security risk. Transport Layer Security (TLS) provides server verification, strong encryption using RSA encryption technology, and other features for protecting data integrity.

Related Information

Transport Layer Security [page 1727]

1.1.6.2.2 IPv6 Support

On IPv6-enabled computers, the network database server listens by default on all IPv4 and IPv6 addresses.

IPv6 is not supported on Windows CE.

Usually no changes are required to the database server start line to use IPv6. In a case where specifying an IP address is required, the database server and the client libraries both accept IPv4 and IPv6 addresses. For example, if a computer has more than one network card enabled, it probably has two IPv4 addresses and two IPv6 addresses. For IPv6 addresses that include a port number, you must enclose the address in either square brackets or parentheses. If you want the database server to listen on only one of the IPv6 addresses, you can specify an address in the following format:

```
dbsrv17 -x tcpip(MyIP=fd77:55f:5a64:52a:202:5445:5245:444f) ...
```

Similarly, if a client application needs to specify the IP address of a server, the connection string or ODBC data source can contain the address, in the following format:

```
...HOST=fd77:55f:5a64::444f;...
```

Each interface is given an interface identifier, which appears at the end of an IPv6 address. For example, if `ipconfig.exe` lists the address `fd77:55f:5a64::444f%7`, the interface identifier is 7. When specifying an IPv6 address on a Windows platform, the interface identifier should be used. On UNIX and Linux, you can specify either an interface identifier or an interface name (the interface name is the name of the interface reported by ifconfig). For example, the interface name is eth1 in the following IPv6 address: `fd77:55f:5a64::444f%eth1`. An interface identifier is required when specifying IPv6 addresses on Linux (kernel 2.6.13 and later). This requirement affects values specified by the following:

- Host connection parameter
- Broadcast (BCAST) protocol option
- Host (IP) protocol option (client side only)
- MyIP (ME) protocol option

Example

Suppose `ipconfig.exe` lists two interfaces, one with the identifier 1 and the other with the identifier 2. If you are looking for a database server that is on the network used by interface number 2, you can tell the client library to broadcast only on that interface:

```
LINKS=tcpip(BROADCAST=ff02::1%2)
```

The IPv6 link-local multicast address is `ff02::1`.

Related Information

Host Connection Parameter [page 86]
Broadcast (BCAST) Protocol Option [page 192]
Host (IP) Protocol Option (Client Side Only) [page 206]
MyIP (ME) Protocol Option [page 229]

1.1.6.2.3 Firewall Connections

There are restrictions on connections when the client application is on the other side of a firewall from the database server.

Firewall software may filter network packets according to network port. Also, it is common to disallow UDP packets from crossing the firewall.

Usually, you can connect through a properly configured firewall using the Host connection parameter and providing the database server address and port. If the database server is using the default port of 2638, the port is not required.

The following connection string fragment connects to a database server named myserver running on a computer at address serverhost using port 2020. No UDP packets are used because the Host connection parameter specifies the TCP/IP address and port.

```
Server=myserver;Host=serverhost:2020
```

Firewalls that Only Allow Certain Client Ports

If the firewall must be configured to allow only certain client ports, then you must use the CommLinks(LINKS) connection parameter instead of the Host connection parameter. The following TCP/IP protocol options are required when you use the CommLinks connection parameter:

Host

Set this protocol option to the host name on which the database server is running. You can use the short form IP.

ServerPort

If your database server is not using the default port of 2638, you must specify the port it is using. You can use the short form PORT.

ClientPort

Set this protocol option to a range of allowed values for the client application to use. You can use the short form CPORT.

DoBroadcast=NONE

Set this protocol option to prevent UDP from being used when connecting to the server. You can use the short from DOBROAD.

Your firewall must be configured to allow TCP/IP traffic between the database server's address and all the clients' addresses. The database server's address is the IP address of the computer running the server (the HOST protocol option) and the database server's IP port number (the ServerPort protocol option, default 2638).

Use a range with more client ports than the maximum number of concurrent connections from each client computer since there is a several minute timeout before a client port can be reused. The range of clients specified in the ClientPort protocol option must match the range allowed by the firewall.

Example

The following connection string fragment restricts the client application to ports 5050 through 5060, and connects to a database server named myeng running on the computer at address myhost using the server port 2020. No UDP broadcast is performed because the DoBroadcast protocol option is set to NONE.

```
Server=myeng;LINKS=tcpip(ClientPort=5050-5060;HOST=myhost;PORT=2020;DoBroadcast=N
ONE)
```

Related Information

CommLinks (LINKS) Connection Parameter [page 59]
ClientPort (CPORT) Protocol Option (Client Side Only) [page 199]
ServerPort (PORT) Protocol Option [page 235]
Host (IP) Protocol Option (Client Side Only) [page 206]
DoBroadcast (DOBROAD) Protocol Option [page 202]

1.1.6.2.4 Dial-up Network Connections (CommLinks Connection Parameter)

To connect over a dial-up network connection, you must use the CommLinks connection parameter.

On the client side, you should specify the following protocol options:

Host

You should specify the host name or IP address of the database server using the Host (IP) protocol option.

DoBroadcast

If you specify the Host (IP) protocol option, there is no need to do a broadcast search for the database server. For this reason, use direct broadcasting.

MyIP

You should set `MyIP=NONE` on the client side.

TIMEOUT

Set the Timeout (TO) protocol option to increase the time the client waits while searching for a server.

Example

The following is an example of a typical CommLinks (LINKS) connection parameter used to connect over a dialup network connection:

```
LINKS=tcpip(MyIP=NONE;DoBroadcast=DIRECT;HOST=server-ip); TIMEOUT=15)
```

Related Information

Host (IP) Protocol Option (Client Side Only) [page 206]
DoBroadcast (DOBROAD) Protocol Option [page 202]
MyIP (ME) Protocol Option [page 229]
Timeout (TO) Protocol Option [page 242]

1.1.6.2.5 Connections Using LDAP as a Name Server

You can specify a central LDAP server to keep track of all database servers in an enterprise if you are operating on a Windows or Unix platform.

When the database server registers itself with an LDAP server, clients can query the LDAP server and find the database server they are looking for, regardless of whether they are on a WAN, a LAN, or going through firewalls. LDAP name server lookup is not utilized by clients that specify the database server address with the Host connection parameter or the HOST protocol option.

In this section:

1.1.6.2.5.1 `saldap.ini` File Configuration

To locate database servers using LDAP as a name server, a file containing information about how to find and connect to the LDAP server must be created on the database server computer and on each client computer.By default the name of this file is `saldap.ini`, but you can rename it.

If you rename the `saldap.ini` file, then you must use the LDAP protocol option to specify the file name when connecting to the database server. If this file doesn't exist, LDAP name server support is disabled.

You can encrypt the contents of the `saldap.ini` file using the File Hiding utility (dbfhide). If you do so, make sure to keep an unencrypted copy of the file so that you will be able to update the encrypted file.

The file must be located in the same directory as the SQL Anywhere executables (for example, *%SQLANY17%* `\bin32` on Microsoft Windows) unless a full path is specified with the LDAP protocol option. The file's contents must be in the following format:

```
[LDAP]
server=computer-running-LDAP-server
port=port-number-of-LDAP-server
basedn=Base-DN
authdn=Authentication-DN
password=password-for-authdn
search_timeout=age-of-timestamps-to-be-ignored
update_timeout=frequency-of-timestamp-updates
read_authdn=read-only-authentication-domain-name
read_password=password-for-authdn
```

server

The name or IP address of the computer running the LDAP server. This value is required on UNIX and Linux. If this entry is missing on Microsoft Windows, then Microsoft Windows looks for an LDAP server running on the local domain controller.

port

The port number used by the LDAP server. The default is 389.

basedn

The domain name of the subtree where the SQL Anywhere entries are stored. This value defaults to the root of the tree.

authdn

The authentication domain name. The domain name must be an existing user object in the LDAP directory that has write access to the basedn. This parameter is required for the database server and it is ignored on the client.

password

The password for authdn. This parameter is required for the database server and it is ignored on the client.

search_timeout

The age at which timestamps are ignored by the client and/or the Server Enumeration utility (dblocate). A value of 0 disables this option so that all entries are assumed to be current. The default is 600 seconds (10 minutes).

update_timeout

The frequency of timestamp updates in the LDAP directory. A value of 0 disables this option so that the database server never updates the timestamp. The default is 120 seconds (2 minutes).

read_authdn

The read-only authentication domain name. The domain name must be an existing user object in the LDAP directory that has read access to the basedn. This parameter is only required if the LDAP server requires a non-anonymous binding before searching can be done. For example, this field is normally required if Active Directory is used as the LDAP server. If this parameter is missing, the bind is anonymous.

read_password

The password for authdn. This parameter is only required on the client if the read_authdn parameter is specified.

How the Connection is Made

When the database server starts, it checks for an existing entry with the same name in LDAP. If a matching entry is found, it is replaced if either the location entries in LDAP match the database server attempting to start, or the timestamp field in the LDAP entry is greater than the time specified by the search_timeout parameter.

If there is another database server running with the same name as the one attempting to start, then the database fails to start.

To ensure that the entries in LDAP are up-to-date, the database server updates a timestamp field in the LDAP entry every 2 minutes. If an entry's timestamp is older than 10 minutes, clients ignore the LDAP entry. Both of these settings are configurable.

On the client, the LDAP directory is searched before the client does any UDP broadcasting, so if the database server is found, no UDP broadcasts are sent. The LDAP search is very fast, so if it fails, there is no discernible delay.

LDAP Server and the Server Enumeration Utility (dblocate)

The Server Enumeration utility (dblocate) also uses LDAP. All database servers listed in LDAP are added to the list of database servers returned. This allows the Server Enumeration utility (dblocate) to list database servers

that wouldn't be returned normally, for example, those that broadcasts wouldn't reach. Entries with timestamps older than 10 minutes are not included.

Example

The following is a sample `saldap.ini` file:

```
[LDAP]
server=ldapserver
basedn=dc=sap,dc=com
authdn=cn=SAServer,ou=iAnywhereASA,dc=sap,dc=com
password=secret
```

The entries are stored in a subtree of the basedn called iAnywhereASA. This entry must be created before SQL Anywhere can use LDAP. To create the subtree, you can use the LDAPADD utility, supplying the following information:

```
dn: ou=iAnywhereASA,basedn
objectClass: organizationalUnit
objectClass: top
ou: iAnywhereASA
```

Related Information

File Hiding Utility (dbfhide) [page 1160]
Server Enumeration Utility (dblocate) [page 1200]

1.1.6.2.5.2 Creating Links in the `/usr/lib` Directory on IBM AIX

Create links in the `/usr/lib` directory, and ensure that the directory containing the LDAP libraries is included in the LIBPATH.

Prerequisites

LDAP is only used with TCP/IP and only with network database servers.

Context

The Server Enumeration utility (dblocate) can use the LDAP server to find other such servers.

Procedure

1. Create links in the `/usr/lib` directory by running the following commands as the root user:

   ```
   cd /usr/lib
   ln -s /opt/IBM/ldap/V6.1/lib64/libibmldap.a libibmldap64.a
   ln -s /opt/IBM/ldap/V6.1/lib/libibmldap.a
   ```

2. (Optional) Ensure that the directory with the LDAP libraries is in the LIBPATH.

   For 32-bit libraries:

   ```
   export LIBPATH=/opt/IBM/ldap/V6.1/lib:$LIBPATH
   ```

   For 64-bit libraries:

   ```
   export LIBPATH=/opt/IBM/ldap/V6.1/lib64:$LIBPATH
   ```

Results

Links are created in `/usr/lib`.

Related Information

LDAP Protocol Option [page 214]
Server Enumeration Utility (dblocate) [page 1200]

1.1.6.2.6 Communication Compression Settings

Enabling compression for one or all connections, and setting the minimum size at which packets are compressed, can improve performance in some circumstances.

Therefore, you should perform a performance analysis on your network with your applications to determine whether changing the compression threshold is beneficial.

Enabling compression increases the quantity of information stored in data packets, thereby reducing the number of packets required to transmit a particular set of data. By reducing the number of packets, the data can be transmitted more quickly.

To determine if enabling compression will help in your particular situation, conduct a performance analysis on your network using your applications before using communication compression in a production environment.

You can tune compression performance by:

Enabling Compression

Enabling compression for a connection (or all connections) can significantly improve performance under some circumstances, including:

- When used over slow networks such as some wireless networks, some modems, serial links, and some WANs.
- When used with SQL Anywhere encryption over a slow network with built-in compression since packets are compressed before they are encrypted.

Enabling compression, however, can sometimes also cause slower performance. For example:

- Communication compression uses more memory and more CPU. It may cause slower performance, especially for LANs and other fast networks.
- Most modems and some slow networks already have built-in compression. In these cases, communication compression will not likely provide additional performance benefits unless you are also encrypting the data.

Modifying the Compression Threshold

> **i** Note
>
> For most networks, the compression threshold does not need to be changed. In rare circumstances, you might adjust the compression threshold to improve performance.
>
> To determine if modifying the compression threshold will help in your particular situation, conduct a performance analysis on your network using your applications before making any adjustments in a production environment.

When compression is enabled, individual packets may or may not be compressed, depending on their size. Packets smaller than the compression threshold are not compressed, even if communication compression is enabled.

Because CPU time is required to compress packets, attempting to compress small packets can actually decrease performance.

In general, lowering the compression threshold value may improve performance on very slow networks, while raising the compression threshold may improve performance by reducing CPU usage. However, since lowering the compression threshold value increases CPU usage on both the client and server, a performance analysis should be done to determine whether changing the compression threshold is beneficial.

Adjusting SQL AnywhereCompression Settings

- Enable communication compression.
  Large data transfers with highly compressible data and larger packet sizes tend to get the best compression rates.
- Adjust the CompressionThreshold setting.
  Lowering the compression threshold value may improve performance on very slow networks, while raising the compression threshold may improve performance by reducing CPU usage.

Related Information

Compress (COMP) Connection Parameter [page 61]
-pc Database Server Option [page 478]

sa_conn_compression_info System Procedure
Compress (COMP) Connection Parameter [page 61]
-pc Database Server Option [page 478]
CompressionThreshold (COMPTH) Connection Parameter [page 63]
-pt Database Server Option [page 482]

1.1.7 Network Protocol Options

Network protocol options enable you to work around traits of different network protocol implementations.

Server side	Client side
For the database server, specify network communication protocol options by using the `-x tcpip` option. For example: `dbsrv17 -x tcpip(PARM1=value1;PARM2=value2;...)` For the database server, specify network transport layer security encryption protocol options by using the -ec tls option. For example: `dbsrv17 -ec tls(PARM1=value1;PARM2=value2;...)` For the database server, specify web server protocol options by using the -xs http, -xs https, or -xs odata option. For example: `dbsrv17 -xs http(PARM1=value1;PARM2=value2;...)`	For client applications, you specify network protocol options using the CommLinks (LINKS) connection parameter: `CommLinks=tcpip(PARM1=value1;PARM2=value2;...)` For client applications, you specify transport layer security encryption options using the Encryption (ENC) connection parameter: `Encryption=tls(PARM1=value1;PARM2=value2;...)`

If there are spaces in an option value, then enclose network protocol options in quotation marks so that they are parsed properly by the system command interpreter. For example:

```
dbisql -c
"host=localhost;server=demo;encryption=tls(fips=yes;certificate_name=RSA
Server;certificate_company=SAP;certificate_unit=SQL
Anywhere;trusted_certificates=rsaroot.crt)"
```

The quotation marks are also required under UNIX and Linux if more than one connection parameter or protocol option is given because these systems interpret the semicolon as a command separator.

Boolean options are turned on with YES, Y, ON, TRUE, T, or 1, and are turned off with any of NO, N, OFF, FALSE, F, or 0. The protocol option names are case-insensitive. For example:

```
dbisql -c "Host=localhost;Server=demo;Encryption=TLS(FIPS=1;Certificate_Name=RSA
Server;Certificate_Company=SAP;Certificate_Unit=SQL
Anywhere;Trusted_Certificates=rsaroot.crt)"
```

You can also include the protocol options in a configuration file and use the *@data* server option to invoke the configuration file.

> **i Note**
>
> Use the Host connection parameter instead of the CommLinks (LINKS) connection parameter. Only use the CommLinks (LINKS) connection parameter if you need to specify TCP/IP options other than Host or ServerPort (PORT).
>
> You cannot specify both CommLinks and Host in a connection string.
>
> When you use the CommLinks (LINKS) connection parameter for a TCP/IP connection, specify a server name by using the ServerName (Server) connection parameter.
>
> If you are connecting to a database server on the same computer as the client, then shared memory is recommended.

TCPIP	HTTP	HTTPS	TLS	OData
Broadcast (BCAST) protocol option	DatabaseName (DBN) protocol option	DatabaseName (DBN) protocol option	certificate_company protocol option (client side only)	ExitOnError (EXIT) protocol option
BroadcastListener (BLISTENER) protocol option (server side only)	KeepaliveTimeout (KTO) protocol option	FIPS protocol option	certificate_name protocol option (client side only)	LogFile (LOG) protocol option
ClientPort (CPORT) protocol option (client side only)	LocalOnly (LO) protocol option	Identity protocol option	certificate_unit protocol option (client side only)	LogVerbosity protocol option
DoBroadcast (DOBROAD) protocol option	LogFile (LOG) protocol option	Identity_Password protocol option	FIPS protocol option	MyIP protocol option
Host (IP) protocol option (client side only)	LogFormat (LF) protocol option	KeepaliveTimeout (KTO) protocol option	Identity protocol option (server side only)	QuietConsole (QUIET) protocol option
LDAP protocol option	LogMaxSize (LSIZE) protocol option	LocalOnly (LO) protocol option	Identity_Password protocol option (server side only)	SecureMyIP protocol option
LocalOnly (LO) protocol option	LogOptions (LOPT) protocol option	LogFile (LOG) protocol option	min_tls_version protocol option (client side only)	SecureServerPort protocol option
MyIP (ME) protocol option	LogRename (LRENAME) protocol option	LogFormat (LF) protocol option	trusted_certificates protocol option (client side only)	ServerPort (PORT) protocol option
ReceiveBufferSize (RCVBUFSZ) protocol option	MaxConnections (MAXCONN) protocol option	LogMaxSize (LSIZE) protocol option	skip_certificate_name_check option (client side only)	SSLKeyStore (KEYSTORE) protocol option
SendBufferSize (SNDBUFSZ) protocol option	MaxRequestSize (MAXSIZE) protocol option	LogOptions (LOPT) protocol option		SSLKeyStorePassword (KEYSTOREPASSWORD) protocol option
ServerPort (PORT) protocol option	MaxRequestVars (MAXVARS) protocol option	MaxConnections (MAXCONN) protocol option		

TCPIP	HTTP	HTTPS	TLS	OData
TDS protocol option (server side only)	MyIP (ME) protocol option	MaxRequestSize (MAXSIZE) protocol option		
Timeout (TO) protocol option	ServerPort (PORT) protocol option	MaxRequestVars (MAXVARS) protocol option		
VerifyServerName (VERIFY) protocol option (client side only)	Timeout (TO) protocol option	MyIP (ME) protocol option		
		min_tls_version protocol option		
		ServerPort (PORT) protocol option		
		Timeout (TO) protocol option		

In this section:

Allows you to use FIPS-certified (FIPS PUB 140-2) security algorithms when encrypting database client/server (TLS) and/or web server (HTTPS) communications.

Host (IP) Protocol Option (Client Side Only) [page 206]
Specifies the address or addresses of the database server.

SecureMyIP Protocol Option [page 209]
Indicates the networking interface on which the database server should listen for secure connections.

identity Protocol Option [page 210]
Specifies the name of an identity file.

identity_password Protocol Option [page 211]
Specifies the password for the encryption certificate.

KeepaliveTimeout (KTO) Protocol Option [page 213]
Specifies the maximum idle time, in seconds, between HTTP requests.

LDAP Protocol Option [page 214]
Allows clients to find database servers without specifying the IP address.

LocalOnly (LO) Protocol Option [page 216]
Allows a database server to restrict connections to the local computer only.

LogFile (LOG) Protocol Option [page 217]
Specifies the name of the file where the database server writes information about web requests.

LogFormat (LF) Protocol Option [page 218]
Controls the format of messages written to the message log file where the database server writes information about web requests, and specifies which fields appear in the messages.

LogMaxSize (LSIZE) Protocol Option [page 220]
Controls the maximum size of the message log file where the database server writes information about web requests.

LogOptions (LOPT) Protocol Option [page 221]
Specifies the types of messages that are recorded in the log where the database server writes information about web requests.

LogRename (LRENAME) Protocol Option [page 223]
Allows users to rename old HTTP debug log files.

LogVerbosity Protocol Option [page 224]
Specifies the verbosity level of the logs.

MaxConnections (MAXCONN) Protocol Option [page 225]
Specifies the maximum number of concurrent HTTP/HTTPS connections accepted by the database server.

MaxRequestSize (MAXSIZE) Protocol Option [page 227]
Specifies the size of the largest request the database server can accept.

MaxRequestVars (MAXVARS) Protocol Option [page 228]
Specifies the maximum number of HTTP input variables allowed in a request that is sent to the database server.

MyIP (ME) Protocol Option [page 229]
Use this option for finer control over which network interface controllers (NIC) or adapters to use when several are present on the database server or client computer.

QuietConsole (QUIET) Protocol Option [page 231]
Tells the OData server whether or not to start with its server messages window minimized.

ReceiveBufferSize (RCVBUFSZ) Protocol Option [page 232]
Sets the size for the receiving buffer used by the TCP/IP protocol stack.

SecureServerPort Protocol Option [page 233]
Specifies the port on which the OData server listens for secure (HTTPS) connections.

SendBufferSize (SNDBUFSZ) Protocol Option [page 234]
Sets the size for the sending buffer used by the TCP/IP protocol stack.

ServerPort (PORT) Protocol Option [page 235]
Specifies the port the database server is listening on.

skip_certificate_name_check Protocol Option (Client Side Only) [page 238]
Controls whether the client library skips the check of the server host name against the database server certificate host names.

SSLKeyStore (KEYSTORE) Protocol Option [page 239]
Specifies the file path for the Java Key Store containing the SSL certificate that the OData server uses to encrypt communication.

SSLKeyStorePassword (KEYSTOREPASSWORD) Protocol Option [page 240]
Specifies the password used by the OData server to authenticate against the Java Key Store identified by the SSLKeyStore (KEYSTORE) protocol option.

TDS Protocol Option (Server Side Only) [page 241]
Controls whether Tabular Data Stream (TDS) connections to a database server are allowed.

Timeout (TO) Protocol Option [page 242]
Specifies the length of time, in seconds, to wait for a response when establishing communications (TCP/IP), and the length of time, in seconds, to wait for a request (HTTP or HTTPS).

trusted_certificate Protocol Option [page 244]
Specifies the path and file name of a file that contains one or more trusted certificates or a PEM-encoded string.

VerifyServerName (VERIFY) Protocol Option (Client Side Only) [page 247]
Controls whether clients must verify the database server name before connecting.

Related Information

Communication Protocol Considerations [page 174]
CommLinks (LINKS) Connection Parameter [page 59]
Encryption (ENC) Connection Parameter [page 80]
Host Connection Parameter [page 86]
ServerName (Server) Connection Parameter [page 110]
-xs Database Server Option [page 529]
CREATE ODATA PRODUCER Statement
ALTER ODATA PRODUCER Statement
DROP ODATA PRODUCER Statement

1.1.7.1 Broadcast (BCAST) Protocol Option

Specifies the IP broadcast address that should be used to find servers.

Syntax

```
{ Broadcast | BCAST }=ip-address
```

Available protocols

TCP/IP

Allowed Values

ip-address

This string must be specified in the form of an IP address.

Default

Broadcasts to all addresses on the same subnet.

Remarks

Database server

When the database server starts, it sends out UDP broadcast packets to see if there is another server out there with the same server name. This option specifies the broadcast address to use for this purpose.

Client

When a client application attempts to locate a server, this option specifies the IP broadcast address that should be used to find the server.

The default broadcast address is created using the local IP address and subnet mask. The subnet mask indicates which portion of the IP address identifies the network, and which part identifies the host.

For example, for a subnet of 10.24.98.x, with a mask of 255.255.255.0, the default broadcast address would be 10.24.98.255.

When specifying an IPv6 address on a Windows platform, the interface identifier should be used. Unix platforms support both interface identifiers and interface names in IPv6 addresses. The interface identifier is required on Linux (kernel 2.6.13 and later).

Example

The following connection string example tells the client to broadcast only on interface number 2 when using IPv6:

```
LINKS=tcpip(BROADCAST=ff02::1%2)
```

The following connection string example tells the client to broadcast all IP addresses in the range 10.24.98.0 to 10.24.98.254 when using IPv4:

```
LINKS=tcpip(BROADCAST=10.24.98.255)
```

Related Information

Troubleshooting: How the Broadcast Repeater Utility Locates Database Servers [page 266]
IPv6 Support [page 178]
BroadcastListener (BLISTENER) Protocol Option (Server Side Only) [page 193]
DoBroadcast (DOBROAD) Protocol Option [page 202]

1.1.7.2 BroadcastListener (BLISTENER) Protocol Option (Server Side Only)

Controls broadcast listening for the specified port.

Syntax

```
{ BroadcastListener | BLISTENER }={ YES | NO }
```

Available protocols

TCP/IP (server side only)

Default

YES

Remarks

This option allows you to turn broadcast listening off for this port.

Using *-sb 0* is the same as specifying `BroadcastListener=NO` on TCP/IP.

If broadcast listening is off, then the database server does not respond to UDP broadcasts. Clients must use the Host connection parameter or HOST protocol option to specify the host name or IP address of the database server, or you must register the database server with LDAP and use LDAP on the clients to find the database server. This also means that the dblocate utility does not include the database server in its output.

Example

The following example starts a database server that accepts TCP/IP connections, and requires that TCP/IP connections use the Host connection parameter or HOST protocol option (when not using LDAP to locate the server).

```
dbsrv17 -x tcpip(BroadcastListener=no) -n myserver ...
```

The following example contains a client connection string that specifies the host name and server name to connect to the database server.

```
dbisql -c "Host=myhost;Server=myserver;UID=DBA;PWD=passwd"
```

If BroadcastListener=yes had been specified on the database server command line, then the connection could have been made specifying the server name, user ID, and password only.

Related Information

-sb Database Server Option [page 490]
Broadcast (BCAST) Protocol Option [page 192]
DoBroadcast (DOBROAD) Protocol Option [page 202]
Server Enumeration Utility (dblocate) [page 1200]
Host Connection Parameter [page 86]
Host (IP) Protocol Option (Client Side Only) [page 206]

1.1.7.3 certificate_company Protocol Option (Client Side Only)

Forces the client to accept server certificates only when the Organization field on the certificate matches this value.

Syntax

```
certificate_company = organization
```

Available protocols

HTTPS, TLS

Default

None

Remarks

Use this option to verify that the Organization field in the identity portion of the certificate matches the value you specify.

When initiating TLS or HTTPS connections, the client libraries check the host name of the database server against the certificate provided by that server using the procedure described in RFC 2818. This check only occurs if none of the certificate_name, certificate_company, or certificate_unit options are specified, or the skip_certificate_name_check option is not enabled. If any of certificate_name, certificate_company, or certificate_unit are specified, then only those options are verified. The skip_certificate_name_check option disables the host name check when enabled.

The host names or IP addresses are derived from the subjectAltName (Subject Alternative Name or SAN) extension and the Common Name (CN) field. The SAN may contain multiple host names with wild cards. For example, a Google certificate might include *.google.com, *.google.ca, and *.android.com. Thus, www.google.ca is a valid host name.

HTTPS is only supported for web services client procedures.

Example

The following command connects Interactive SQL to the database server named demo by using transport layer security.

```
dbisql -c "UID=DBA;PWD=sql;Server=demo;Host=localhost;Encryption=
  TLS(fips=yes;trusted_certificate=rsaroot.crt;certificate_name=RSA
Server;certificate_company=SAP;certificate_unit=SQL Anywhere)"
```

Related Information

SQL Anywhere Client/Server Communication Encryption [page 1745]
Separately Licensed Components
certificate_name Protocol Option (Client Side Only) [page 196]
certificate_unit Protocol Option (Client Side Only) [page 198]
trusted_certificate Protocol Option [page 244]
Encryption (ENC) Connection Parameter [page 80]
Certificate Creation Utility (createcert) [page 1130]
CREATE PROCEDURE Statement [Web Service]

1.1.7.4 certificate_name Protocol Option (Client Side Only)

Forces the client to accept server certificates only when the Common Name field on the certificate matches this value.

Syntax

```
certificate_name=common-name
```

Available protocols

HTTPS, TLS

Default

None

Remarks

Use this option to verify that the Common Name field in the identity portion of the certificate matches the value you specify.

When initiating TLS or HTTPS connections, the client libraries check the host name of the database server against the certificate provided by that server using the procedure described in RFC 2818. This check only occurs if none of the certificate_name, certificate_company, or certificate_unit options are specified, or the skip_certificate_name_check option is not enabled. If any of certificate_name, certificate_company, or certificate_unit are specified, then only those options are verified. The skip_certificate_name_check option disables the host name check when enabled.

The host names or IP addresses are derived from the subjectAltName (Subject Alternative Name or SAN) extension and the Common Name (CN) field. The SAN may contain multiple host names with wild cards. For example, a Google certificate might include *.google.com, *.google.ca, and *.android.com. Thus, www.google.ca is a valid host name.

HTTPS is only supported for web services client procedures.

Example

The following command connects Interactive SQL to the database server named demo by using transport layer security.

```
dbisql -c "UID=DBA;PWD=sql;Server=demo;Host=localhost;Encryption=
  TLS(fips=yes;trusted_certificate=rsaroot.crt;certificate_name=RSA
Server;certificate_company=SAP;certificate_unit=SQL Anywhere)"
```

Related Information

SQL Anywhere Client/Server Communication Encryption [page 1745]
Separately Licensed Components
certificate_company Protocol Option (Client Side Only) [page 195]
certificate_unit Protocol Option (Client Side Only) [page 198]
trusted_certificate Protocol Option [page 244]
Encryption (ENC) Connection Parameter [page 80]
Certificate Creation Utility (createcert) [page 1130]
CREATE PROCEDURE Statement [Web Service]

1.1.7.5 certificate_unit Protocol Option (Client Side Only)

Forces the client to accept server certificates only when the Organization Unit field on the certificate matches this value.

Syntax

```
certificate_unit=organization-unit
```

Available protocols

HTTPS, TLS

Default

None

Remarks

Use this option to verify that the Organization Unit field in the identity portion of the certificate matches the value you specify.

When initiating TLS or HTTPS connections, the client libraries check the host name of the database server against the certificate provided by that server using the procedure described in RFC 2818. This check only occurs if none of the certificate_name, certificate_company, or certificate_unit options are specified, or the skip_certificate_name_check option is not enabled. If any of certificate_name, certificate_company, or certificate_unit are specified, then only those options are verified. The skip_certificate_name_check option disables the host name check when enabled.

The host names or IP addresses are derived from the subjectAltName (Subject Alternative Name or SAN) extension and the Common Name (CN) field. The SAN may contain multiple host names with wild cards. For example, a Google certificate might include *.google.com, *.google.ca, and *.android.com. Thus, www.google.ca is a valid host name.

HTTPS is only supported for web services client procedures.

Example

The following command connects Interactive SQL to the server demo using transport layer security.

```
dbisql -c "UID=DBA;PWD=sql;Server=demo;Host=localhost;Encryption=
```

```
   TLS(fips=yes;trusted_certificate=rsaroot.crt;certificate_name=RSA
Server;certificate_company=test;certificate_unit=test)"
```

Related Information

SQL Anywhere Client/Server Communication Encryption [page 1745]
Separately Licensed Components
certificate_company Protocol Option (Client Side Only) [page 195]
certificate_name Protocol Option (Client Side Only) [page 196]
trusted_certificate Protocol Option [page 244]
Encryption (ENC) Connection Parameter [page 80]
Certificate Creation Utility (createcert) [page 1130]
CREATE PROCEDURE Statement [Web Service]

1.1.7.6 ClientPort (CPORT) Protocol Option (Client Side Only)

Designates the port number on which the client application communicates using TCP/IP.

Syntax

```
{ ClientPort | CPORT }=port-number
```

Available protocols

TCP/IP (client side only)

Default

Assigned dynamically per connection by the networking implementation. If you do not have firewall restrictions, it is recommended that you do not use this protocol option.

Remarks

This option is provided for connections across firewalls, as firewall software filters according to TCP/UDP port. It is recommended that you do not use this protocol option unless you need to for firewall reasons.

The ClientPort option designates the port number on which the client application communicates using TCP/IP. You can specify a single port number, or a combination of individual port numbers and ranges of port numbers. For example:

- (CPORT=1234)
- (CPORT=1234,1235,1239)
- (CPORT=1234-1238)
- (CPORT=1234-1237,1239,1242)

It is best to specify a list or a range of port numbers to make multiple connections using a given Data Source or a given connection string. If you specify a single port number, then your application will be able to maintain only one connection at a time. In fact, even after closing the one connection, there is a several minute timeout period during which no new connection can be made using the specified port. When you specify a list and/or range of port numbers, the application keeps trying port numbers until it finds one to which it can successfully bind.

Example

The following connection string fragment makes a connection from an application using port 6000 to a database server named my-server using port 5000:

```
CommLinks=tcpip(ClientPort=6000;ServerPort=5000);ServerName=my-server
```

The following connection string fragment makes a connection from an application that can use the specified ports for communicating with a database server named my-server using the default server port:

```
CommLinks=tcpip(ClientPort=5040,5050-5060,5070);
ServerName=my-server
```

Related Information

Communication Protocols [page 173]
Firewall Connections [page 179]
Host (IP) Protocol Option (Client Side Only) [page 206]
DoBroadcast (DOBROAD) Protocol Option [page 202]
ServerPort (PORT) Protocol Option [page 235]

1.1.7.7 DatabaseName (DBN) Protocol Option

Specifies the name of a database to use when processing web requests, or uses the REQUIRED or AUTO keyword to specify whether database names are required as part of the URL.

Syntax

```
{ DatabaseName | DBN }={ AUTO | REQUIRED | database-name }
```

Available protocols

HTTP, HTTPS

Default

AUTO

Remarks

If this protocol option is set to REQUIRED, the URL must specify a database name.

If this protocol option is set to AUTO, the URL may specify a database name, but does not need to do so. If the URL contains no database name, the default database on the server is used to process web requests. Since the database server must determine whether the URL contains a database name when set to AUTO, you should avoid ambiguity in your web site design.

If this protocol option is set to the name of a database, that database is used to process all web requests. The URL must not contain a database name.

Example

The following command starts two databases, but permits only one of them to be accessed via HTTP.

```
dbsrv17 -xs http(DBN=web) "%SQLANYSAMP17%\demo.db" web.db
```

The following command starts two HTTP web services - one for `your-first-database.db` and one for `your-second-database.db`:

```
dbsrv17 -xs http(PORT=80;DBN=your-first-database),http(PORT=8800;DBN=your-second-database) your-first-database.db your-second-database.db
```

1.1.7.8 DoBroadcast (DOBROAD) Protocol Option

Controls how a client searches for a database server, and controls whether the database server broadcasts when it starts.

Syntax

```
(database server)
  { DoBroadcast | DOBROAD }={ YES | NO }
(client)
  { DoBroadcast | DOBROAD }={ ALL | NONE | DIRECT }
```

Available protocols

TCP/IP

Allowed values (database server)

YES

Setting DoBroadcast=YES allows the database server to broadcast to find other database servers with the same name when it starts.

NO

Setting DoBroadcast=NO prevents the database server from broadcasting to find other database servers with the same name when starting up. This is useful in certain rare circumstances, but it is not generally recommended.

Allowed values (client)

ALL

With DoBroadcast=ALL a broadcast is performed to search for a database server. The broadcast goes first to the local subnet. If HOST is specified, broadcast packets are also sent to each of the hosts. All broadcast packets are UDP packets.

NONE

Specifying DoBroadcast=NONE causes no UDP broadcasts to be used, and the database server address cache (located in the `sasrv.ini` file) is ignored. A TCP/IP connection is made directly with the HOST/PORT specified, and the database server name is verified. If you specify DoBroadcast=NONE, the HOST (IP) protocol option is required.

DIRECT

With DoBroadcast=DIRECT, no broadcast is performed to the local subnet to search for a database server. Broadcast packets are sent only to the hosts listed in the HOST (IP) protocol option. If you specify DoBroadcast=DIRECT, the HOST (IP) protocol option is required.

Default

YES (database server)

ALL (client) If the HOST (IP) protocol option is specified, the default is DIRECT.

Remarks

Client usage

With TCP/IP, you can choose not to verify the database server name by setting the VerifyServerName (VERIFY) protocol option to NO. The HOST (IP) protocol option is a required parameter, unless LDAP is being used, while the ServerPort (PORT) protocol option is optional.

For DIRECT and NONE, you must specify the database server host with the HOST option.

Example

The following connection string fragment connects to a database server named gold running on the computer named silver.

```
Host=silver;ServerName=gold
```

The following connection string fragment uses the CommLinks protocol option to connect to the same database server.

```
CommLinks=tcpip(DOBROADCAST=DIRECT;HOST=silver);ServerName=gold
```

Related Information

Broadcast (BCAST) Protocol Option [page 192]
BroadcastListener (BLISTENER) Protocol Option (Server Side Only) [page 193]

1.1.7.9 ExitOnError (EXIT) Protocol Option

Controls whether the OData server shuts down if an error occurs on startup.

Syntax

```
{ ExitOnError | EXIT }={ YES | NO }
```

Available protocols

OData

Allowed values (database server)

YES

Setting ExitOnError=YES tells the OData server to shut down if an error occurs on startup.

NO

Setting ExitOnError=NO tells the OData server to remain running if it an error occurs on startup.

Default

YES

Remarks

Specify whether the OData server shuts down if an error occurs on startup.

Example

Start a database server that listens on port 8080 for OData requests and that shuts down if an error occurs on startup:

```
dbsrv17 -xs odata(ServerPort=8080;EXIT=YES) c:\mydatabase.db
```

1.1.7.10 FIPS Protocol Option

Allows you to use FIPS-certified (FIPS PUB 140-2) security algorithms when encrypting database client/server (TLS) and/or web server (HTTPS) communications.

Syntax

```
FIPS={ YES | NO }
```

Available protocols

HTTPS, TLS

Allowed Values

YES, NO

Default

NO

Remarks

FIPS-certified encryption is only supported for RSA encryption. Whether the code used to implement RSA encryption is certified or not, the same encryption takes place. Clients that use the non-FIPS-certified encryption algorithms can communicate with servers that use FIPS-certified encryption algorithms and vice versa.

FIPS-certified encryption requires a separate license.

Example

The following command line configures the ServerPort, FIPS, Identity, and Identity_Password network protocol options for a web server.

```
dbsrv17 -xs
https(serverport=544;fips=yes;identity=rsaserver.id;identity_password=test)
demo.db
```

The following command line configures the FIPS, Identity, and Identity_Password network protocol options for a database server.

```
dbsrv17 -ec tls(fips=yes;identity=rsaserver.id;identity_password=test) demo.db
```

The following command connects Interactive SQL to the database server demo, started in the previous example, using transport layer security.

```
dbisql -c "UID=DBA;PWD=sql;Server=demo;Host=localhost;Encryption=
  TLS(fips=yes;trusted_certificate=rsaroot.crt;certificate_name=RSA
Server;certificate_company=SAP;certificate_unit=SQL Anywhere)"
```

The certificate_name, certificate_company, and certificate_unit options restrict connections to a server using the `rsaserver.id` certificate file.

Related Information

SQL Anywhere Client/Server Communication Encryption [page 1745]
Separately Licensed Components
certificate_company Protocol Option (Client Side Only) [page 195]
certificate_name Protocol Option (Client Side Only) [page 196]
certificate_unit Protocol Option (Client Side Only) [page 198]
trusted_certificate Protocol Option [page 244]
Encryption (ENC) Connection Parameter [page 80]
Certificate Creation Utility (createcert) [page 1130]
-ec Database Server Option [page 418]
-fips Database Server Option [page 426]
-xs Database Server Option [page 529]

1.1.7.11 Host (IP) Protocol Option (Client Side Only)

Specifies the address or addresses of the database server.

Syntax

```
{ Host | IP }=ip-address
```

Available protocols

TCP/IP (client side only)

Allowed Values

ip-address

This string must be specified in the form of an IP address and it can optionally include a port number (separated by a colon). The list of host values is a comma-separated list. For IPv6 addresses that include a port number, you must enclose the address in either square brackets or parentheses. You can use **localhost** to identify the current computer. For example:

```
links=tcpip(HOST=myhost)
links=tcpip(HOST=myhost:1234)
links=tcpip(HOST=10.25.13.5,myotherhost)
links=tcpip(HOST=myhost:1234,10.25.65.112)
links=tcpip(HOST=myhost:1234,myotherhost:5678)
```

Default

Search the current TCP/IP subnet.

Remarks

Host specifies the address or addresses of the database server. IP and Host are synonyms. You can use a comma-separated list of addresses to search for more than one computer. You can also append a port number to an IP address, using a colon as separator. Alternatively, you can specify the host and server ports explicitly using the ServerPort protocol option, as in `HOST=myhost;PORT=5000`. For IPv6 addresses that include a port number, you must enclose the address in either square brackets or parentheses, for example `[fe80::5445:5245:444f]:2638`. If a port is specified, only that port number is used for TCP/IP connections and UDP broadcasts. If a port number is not specified, port 2638 is used.

When specifying an IPv6 address on a Microsoft Windows platform, the interface identifier should be used. UNIX and Linux platforms support both interface identifiers and interface names in IPv6 addresses. The interface identifier is required on Linux (kernel 2.6.13 and later).

You must use the host name **localhost** when connecting to a personal database server using TCP/IP. You must enable the TCP/IP communication protocol on the personal database server using the -x option.

Specify the LogFile connection parameter to have the client application write the addressing information to its log file.

You cannot specify the LocalOnly protocol option with the Host protocol option or the Host connection parameter.

The Host protocol option is different from the Host connection parameter. The Host protocol option is used with the CommLinks connection parameter. You should only use the CommLinks (LINKS) connection parameter if you need to specify TCP/IP options other than Host or ServerPort (PORT). You cannot specify both CommLinks and Host in the connection string.

The Host connection parameter must be specified when using TLS encryption and the host name must match the Common Name in the database server's identity certificate, unless the skip_certificate_name_check option

is enabled in the Encryption (ENC) connection parameter. The Host protocol option cannot be used as an alternative in this case.

The HOST protocol option is a hint to the client library of where the server might be. It's still possible to find a server on a different host.

Example

The following connection string fragment instructs the client to look on the computers kangaroo and 197.75.209.222 (port 2369) to find a database server:

```
LINKS=tcpip(IP=kangaroo,197.75.209.222:2369)
```

The following connection string fragment instructs the client to look on the computers my-server and kangaroo to find a database server.

```
LINKS=tcpip(HOST=my-server,kangaroo;PORT=2639)
```

The following connection string fragment instructs the client to look for a database server on host1 running on port 1234 and for a database server on host2 running on port 4567. The client does not look on host1 on port 4567 or on host2 on port 1234.

```
LINKS=tcpip(HOST=host1:1234,host2:4567)
```

The following connection string fragment instructs the client to look for a database server on an IPv6 address:

```
LINKS=tcpip(HOST=fe80::5445:5245:444f)
```

The following connection string fragment instructs the client to look for a database server on an IPv6 address on port 2639:

```
LINKS=tcpip(HOST=[fe80::5445:5245:444f]:2639)
```

The following examples demonstrate using IPv6 addresses with the Host protocol option:

```
// Global scope address, unique everywhere, so no interface index is required
// no index required
-c "links=tcpip(host=fd77:55d:59d9:56a:202:55ff:fe76:df19)"
// all communication is done through interface 2
-c "links=tcpip(host=fd77:55d:59d9:56a:202:55ff:fe76:df19%2)"
// all communication is done through eth0
-c "links=tcpip(host=fd77:55d:59d9:56a:202:55ff:fe76:df19%eth0)"
// Link scope address, addresses are unique on each interface
// possibly ambiguous (this host may exist through both eth0 and eth1)
-c "links=tcpip(host=fe80::202:55ff:fe76:df19)"
// not ambiguous because it must use interface 2
-c "links=tcpip(host=fe80::202:55ff:fe76:df19%2)"
// not ambiguous because it must use eth0
-c "links=tcpip(host=fe80::202:55ff:fe76:df19%eth0)"
```

Related Information

CommLinks (LINKS) Connection Parameter [page 59]
Host Connection Parameter [page 86]
IPv6 Support [page 178]
-z Database Server Option [page 532]
LogFile (LOG) Connection Parameter [page 97]
ClientPort (CPORT) Protocol Option (Client Side Only) [page 199]
ServerPort (PORT) Protocol Option [page 235]

1.1.7.12 SecureMyIP Protocol Option

Indicates the networking interface on which the database server should listen for secure connections.

Syntax

```
SecureMyIP=IP-address
```

Available protocols

OData

Allowed Values

IP-address

This string must be specified in the form of an IP address.

Remarks

The SecureMyIP protocol option is provided for computers with more than one network adapter.

Each adapter has an IP address. By default, the database server uses every network interface it finds. If you don't want your database server to listen on all network interfaces, specify the address of the interface you want to use in the SecureMyIP protocol option.

When specifying an IPv6 address on a Windows platform, use the interface identifier. Unix platforms support both interface identifiers and interface names in IPv6 addresses. The interface identifier is required on Linux (kernel 2.6.13 and later).

Example

The following command instructs the database server to listen for secure OData requests on the specified secure network and non-secure requests on all interfaces:

```
dbsrv17 -xs odata(SecureMyIP=192.75.209.12;SSLKeyStore=c:
\mykeys.jks;SSLKeyStorePassword=password)
```

1.1.7.13 identity Protocol Option

Specifies the name of an identity file.

Syntax

```
identity=identity-file
```

Available protocols

HTTPS, TLS

Allowed Values

identity-file

This string specifies the name of an identity file.

Default

There is no default identity file name.

Remarks

When you use transport layer security, the identity file contains the public certificate and its private key. For certificates that are not self signed, the identity file also contains all the signing certificates. Specify the password for the private key by using the identity_password option.

Example

Start a database server that requires client connections to use TLS encryption.

```
dbsrv17 -ec tls(fips=no;identity=rsaserver.id;identity_password=test) demo.db
```

The following command connects Interactive SQL to the database server demo, started in the previous example, using transport layer security.

```
dbisql -c "UID=DBA;PWD=sql;Server=demo;Host=localhost;Encryption=
  TLS(FIPS=yes;trusted_certificate=rsaroot.crt;certificate_name=RSA
Server;certificate_company=SAP;certificate_unit=SQL Anywhere)"
```

Start a database server that requires web connections to use TLS encryption.

```
dbsrv17 -xs https(identity=rsaserver.id;identity_password=test) demo.db
```

> ⚠ Caution
>
> The sample identity file is intended for use only during testing and development. It provides no protection because it is a standard part of SQL Anywhere. Replace it with your own certificate before deploying your application.

Start a client application that requires the database server to use TLS encryption:

```
dbping -d -c
"UID=DBA;PWD=sql;HOST=myserverhost;SERVER=myservername;DBN=mydatabase;ENC=TLS(tru
sted_certificate=myroot.crt; identity=myclient.id;identity_password=client-
password)"
```

Related Information

Using Client-side Certificates for TLS [page 1744]
How to Set up Transport Layer Security [page 1731]
identity_password Protocol Option [page 211]
trusted_certificate Protocol Option [page 244]
Encryption (ENC) Connection Parameter [page 80]
-ec Database Server Option [page 418]

1.1.7.14 identity_password Protocol Option

Specifies the password for the encryption certificate.

> ⁖ Syntax
>
> *identity_password*=password

Available protocols

HTTPS, TLS

Allowed Values

password

This string specifies the password for an encryption certificate.

Default

There is no default identity file password.

Remarks

When you use transport layer security, this option specifies the password that matches the password for the encryption certificate specified by the identity protocol option.

Example

Start a database server that requires web connections to use a particular certificate.

```
dbsrv17 -xs https(identity=rsaserver.id;identity_password=test) demo.db
```

Start a database server that requires client connections to use a particular certificate.

```
dbsrv17 -ec tls(identity=rsaserver.id;identity_password=test) demo.db
```

Start a client application that provides a verifiable certificate to the database server and requires the database server to use a particular certificate:

```
dbping -d -c
"UID=DBA;PWD=sql;HOST=myserverhost;SERVER=myservername;DBN=mydatabase;ENC=TLS(tru
sted_certificate=myroot.crt; identity=myclient.id;identity_password=client-
password)"
```

Related Information

Using Client-side Certificates for TLS [page 1744]
How to Set up Transport Layer Security [page 1731]
identity Protocol Option [page 210]
trusted_certificate Protocol Option [page 244]
Encryption (ENC) Connection Parameter [page 80]
-ec Database Server Option [page 418]

1.1.7.15 KeepaliveTimeout (KTO) Protocol Option

Specifies the maximum idle time, in seconds, between HTTP requests.

Syntax

```
{ KeepaliveTimeout | KTO }=timeout-value
```

Available protocols

HTTP

HTTPS

Allowed Values

timeout-value

This integer specifies the length of time, in seconds, to wait between HTTP requests. If you do not want the connection to time out, specify KTO=0. In production environments, it is not recommended to set KTO=0 because, a rogue client could consume the database server's resources and prevent other clients from connecting.

Default

60

Remarks

This protocol option only applies to keep-alive HTTP requests. It specifies the maximum idle time permitted from the time the last HTTP request was received to the time when data is first received for the next HTTP request. Once data is received for a new request, then the timeout period, as defined by the Timeout protocol option, takes effect.

Example

```
dbsrv17 -n demo1 -xs https(port=443;
identity=C:\Users\Public\Documents\SQL Anywhere
         17\Samples\Certificates\rsaserver.id;
identity_password=test;kto=50) demo.db
```

Related Information

Timeout (TO) Protocol Option [page 242]

1.1.7.16 LDAP Protocol Option

Allows clients to find database servers without specifying the IP address.

Syntax

```
LDAP={ YES | NO | filename }
```

Usage

TCP/IP

Allowed Values

YES

Specifying LDAP=YES enables LDAP name server support and uses `saldap.ini` (the default file name) as the configuration file.

NO

Specifying NO disables LDAP name server support.

filename

Specifying LDAP=`filename` enables LDAP name server support and uses the specified file as the configuration file.

Default

YES

Remarks

Having the database server register itself with an LDAP server allows clients to query the LDAP server. This allows clients running over a WAN or through a firewall to find database servers without specifying the IP address. It also allows the Locate utility (dblocate) to find such servers.

Unless a full path is specified with the LDAP protocol option, the `saldap.ini` file must be located in the same directory as the SQL Anywhere executables (for example, *%SQLANY17%*`\bin32` on Windows).

You can encrypt the contents of the `saldap.ini` file using the File Hiding utility (dbfhide). If you do so, make sure to keep an unencrypted copy of the file so that you will be able to update the encrypted file.

Using LDAP as a name server is only supported for TCP/IP server communication.

Example

The following example starts a server with LDAP

```
dbsrv17 -n demo17 -x tcpip(ldap=saldap.ini) demo.db
```

The following example connects to the LDAP server

```
dbisql -c links=tcpip(ldap=y);servername=demo17;uid=dba;pwd=sql
```

Related Information

Connections Using LDAP as a Name Server [page 181]
saldap.ini File Configuration [page 182]
Hiding the Contents of an .ini File [page 591]

1.1.7.17 LocalOnly (LO) Protocol Option

Allows a database server to restrict connections to the local computer only.

Allows a client to choose to connect only to a database server on the local computer, if one exists.

Syntax

```
{ LocalOnly | LO }={ YES | NO }
```

Available protocols

TCP/IP, HTTP, HTTPS

Default

NO

Remarks

Database server

You can use the LocalOnly (LO) protocol option with the database server to restrict connections to the local computer. Connection attempts from remote computers will not find this server, and the Locate (dblocate) utility will not see this server.

The default value is NO, which means the database server accepts requests from clients no matter where they are located.

When set to YES, this option causes a network database server to reject all connections from clients running on different computers.

When set to YES the network server runs as a personal server without experiencing connection or CPU limits. This option has no effect on personal database servers, which never accept requests from other computers.

Client

You can use the LocalOnly (LO) protocol option to restrict connections to a database server running on the local computer.

The default value is NO, which means the client will attempt to connect to a server no matter where it is located.

When set to YES, the regular broadcast mechanism is still used but broadcast responses from database servers on other computers are ignored.

You cannot specify the LocalOnly protocol option with the HOST protocol option or the Host connection parameter.

The LocalOnly (LO) protocol option is only useful and accepted if DoBroadcast=ALL (the default) is also specified.

Related Information

How to Start an HTTP Web Server
Broadcast (BCAST) Protocol Option [page 192]
DoBroadcast (DOBROAD) Protocol Option [page 202]

1.1.7.18 LogFile (LOG) Protocol Option

Specifies the name of the file where the database server writes information about web requests.

Syntax

```
{ LogFile | LOG }=filename
```

Available protocols

HTTP, HTTPS, OData

Default

None

Remarks

This protocol option specifies the name of the file to which the database server writes information about web requests.

Example

Start a database server that listens on port 8080 for OData requests and uses the log file `myodata.log`:

```
dbsrv17 -xs odata(ServerPort=8080;LOG=myodata.log) c:\mydatabase.db
```

Related Information

LogFormat (LF) Protocol Option [page 218]
LogMaxSize (LSIZE) Protocol Option [page 220]
LogOptions (LOPT) Protocol Option [page 221]

1.1.7.19 LogFormat (LF) Protocol Option

Controls the format of messages written to the message log file where the database server writes information about web requests, and specifies which fields appear in the messages.

Syntax

```
{ LogFormat | LF }=format-string
```

Available protocols

HTTP, HTTPS

Allowed Values

format-string

The following codes are supported:

@@

The @ character.

@B

Date and time that processing of the request started, unless the request could not be queued due to an error.

@C

Date and time that the client connected.

@D

Name of the database associated with the request.

@E

Text of the error message, if an error occurred.

@F

Date and time that processing of the request finished.

@I

IP address of the client.

@J

Log the client port specified by the @I option.

@L

Length of the response, in bytes, including headers and body.

@M

HTTP request method.

@P

Listener port associated with the request.

@Q

Date and time that the request was queued for processing, unless the request could not be queued due to an error.

@R

Status code and description of the HTTP response.

@S

HTTP status code.

@T

Date and time that the current log entry was written.

@U

Requested URI.

@V

Requested HTTP version.

@W

Time taken to process the request (@F - @B), or 0.000 if the request was not processed due to an error.

Default

@T - @W - @I:@J - @P - "@M @U @V" - @R - @L - @E

Remarks

This protocol option controls the format of messages written to the message log file that stores information about web requests and which fields appear in them. If they appear in the string, the current values are substituted for the codes as each message is written.

If the web request fails due to an unsupported HTTP request method or a URI that is either malformed or missing a required database name, the HTTP method (@M) and HTTP version (@V) returns the string *???* and the URI (@U) returns the given request preceded by >>>.

For example, if the log format is set to "@M @U @V", then an unknown HTTP method for the URL `request/sample/test HTTP/1.0` returns the following:

```
"???>>>request/sample/test HTTP/1.0???"
```

Related Information

LogFile (LOG) Protocol Option [page 217]
LogMaxSize (LSIZE) Protocol Option [page 220]
LogOptions (LOPT) Protocol Option [page 221]

1.1.7.20 LogMaxSize (LSIZE) Protocol Option

Controls the maximum size of the message log file where the database server writes information about web requests.

Syntax

```
{ LogMaxSize | LSIZE }=size[ k | m | g ]
```

Available protocols

HTTP, HTTPS

Allowed Values

size

This integer specifies the maximum size of the file where web request information is written. The default value is in bytes, but you can use k, m, or g to specify units of kilobytes, megabytes, or gigabytes, respectively. If LogMaxSize is zero, the message log file size is unlimited.

Default

0

Remarks

When the message log file reaches the stated size, it is renamed and another log file is created.

Related Information

LogFile (LOG) Protocol Option [page 217]
LogFormat (LF) Protocol Option [page 218]
LogOptions (LOPT) Protocol Option [page 221]

1.1.7.21 LogOptions (LOPT) Protocol Option

Specifies the types of messages that are recorded in the log where the database server writes information about web requests.

Syntax

```
{ LogOptions | LOPT }= [ NONE ] [, OK ] [, INFO ] [, ERRORS ] [, ALL ] [, status-codes ] [, REQHDRS ] [, RESHDRS ] [, HEADERS ]
```

Available protocols

HTTP, HTTPS

Allowed Values

The following keywords control which categories of messages are logged:

NONE

Log nothing.

OK

Log requests that complete successfully (20x HTTP status codes).

INFO

Log requests that return over or not modified status codes (3xx HTTP status codes).

ERRORS

Log all errors (4xx and 5xx HTTP status codes).

USER

Log all errors (6xx, 7xx, 8xx, and 9xx HTTP status codes).

ALL

Log all requests.

The following common HTTP status codes are also available. They can be used to log requests that return particular status codes:

C200

OK

C400

Bad request

C401

Unauthorized

C403

Forbidden

C404

Not found

C408

Request timeout

C501

Not implemented

C503

Service unavailable

The following keywords can be used to obtain more information about the logged messages:

REQHDRS

When logging requests, also write request headers to the message log file.

RESHDRS

When logging requests, also write response headers to the message log file.

HEADERS

When logging requests, also write both request and response headers to the message log file (same as REQHDRS,RESHDRS).

Default

ALL

Remarks

The values available include keywords that select particular types of messages, and HTTP status codes. Multiple values may be specified, separated by commas.

Related Information

LogFile (LOG) Protocol Option [page 217]
LogFormat (LF) Protocol Option [page 218]
LogMaxSize (LSIZE) Protocol Option [page 220]

1.1.7.22 LogRename (LRENAME) Protocol Option

Allows users to rename old HTTP debug log files.

Syntax

```
{ LogRename | LRENAME } = { OLD | DATE | number }
```

Available protocols

HTTP

Allowed Values

OLD Renames the existing HTTP debug log file to `logfile.old` and deletes any existing `logfile.old` files. This is the default.

DATE

Renames the existing HTTP debug log file to `logfile-date-num.ext`, where `logfile.ext` is the original HTTP debug log file, `date` is the current date in the format YYMMDD, and `num` is a two-digit number starting at 00.

> **i** Note
>
> If you specify LogRename=DATE, then an unlimited number of HTTP debug log files may be created. Clean up unused files as necessary.

number Keeps the specified number of old HTTP debug log files and names them `logfile.1`, `logfile.2`, `logfile.3`, and so on. `number` must be a positive integer.

Default

OLD

Remarks

When used with the Log and LogMaxSize protocol options, LogRename tells the database server how to rename old HTTP debug log files.

1.1.7.23 LogVerbosity Protocol Option

Specifies the verbosity level of the logs.

> Syntax
>
> *LogVerbosity* ={ *1* | *2* | *3* | *4* }

Available protocols

OData

Allowed Values

1

Outputs information about any unexpected errors.

2

Outputs general information and configuration messages.

3

Outputs detailed information about HTTP requests.

4

Outputs debugging messages.

Default

2

Remarks

Higher verbosity levels log additional information and include the information provided by all lower levels.

Example

Start a database server that listens on port 8080 for OData requests with the log verbosity level set to 3:

```
dbsrv17 -xs odata(ServerPort=8080;LogVerbosity=3) c:\mydatabase.db
```

1.1.7.24 MaxConnections (MAXCONN) Protocol Option

Specifies the maximum number of concurrent HTTP/HTTPS connections accepted by the database server.

Syntax

```
{ MaxConnections | MAXCONN }=size
```

Available protocols

HTTP, HTTPS

Allowed Values

size

This integer specifies the number of concurrent connections accepted by the database server. The value 0 indicates no limit.

Default

7 (personal server)

Number of licensed connections (network server)

Remarks

The personal server can accept a maximum of ten concurrent connections, of which only seven can be HTTP/HTTPS connections.

The database server queues HTTP/HTTPS connection attempts when the database server reaches its HTTP/HTTPS connection limit. While there are connections in the HTTP/HTTPS queue, there is no opportunity for a standard connection to replace an HTTP/HTTPS connection. A user wanting to make a standard connection could wait indefinitely unless the number of HTTP/HTTPS connections is limited or some of the database connections are reserved for standard connections. The network database server does not reserve connections for standard connection. Reserve standard connections by using the reserved_connections database option.

Related Information

MaxRequestSize (MAXSIZE) Protocol Option [page 227]
MaxRequestVars (MAXVARS) Protocol Option [page 228]
Server Licensing Utility (dblic) [page 1203]
RESERVED_CONNECTIONS Option [page 811]

1.1.7.25 MaxRequestSize (MAXSIZE) Protocol Option

Specifies the size of the largest request the database server can accept.

Syntax

```
{ MaxRequestSize | MAXSIZE }=size[ k | m | g ]
```

Available protocols

HTTP, HTTPS

Allowed Values

size

This integer specifies the size of the largest request the database server can accept. The default value is in bytes, but you can use k, m, or g to specify units of kilobytes, megabytes, or gigabytes, respectively. The value 0 disables this limit, but should be used with extreme caution. Without this limit, a rogue client could overload the database server or cause it to run out of memory.

Default

100k

Remarks

If the size of a request exceeds this limit, the connection is closed and the server returns a response to the client indicating that the request is too large. The message returned is HTTP status code 413 with text "Request Entity Too Large".

Example

The following command instructs the database server to accept requests up to 150000 bytes in size:

```
dbsrv17 -xs http(MaxRequestSize=150000)
```

Related Information

MaxConnections (MAXCONN) Protocol Option [page 225]
MaxRequestVars (MAXVARS) Protocol Option [page 228]

1.1.7.26 MaxRequestVars (MAXVARS) Protocol Option

Specifies the maximum number of HTTP input variables allowed in a request that is sent to the database server.

Syntax

```
{ MaxRequestVars | MAXVARS }=number
```

Available protocols

HTTP, HTTPS

Allowed Values

number

This integer specifies the maximum number of input variables allowed in a request sent to the database server. The value 0 indicates that there is no upper limit; however, the number of input variables allowed is still limited by the number of bytes allowed in the request, which is determined by the MaxRequestSize protocol option.

Default

10000

Remarks

If the number of input variables exceeds this limit, the connection is closed and the client returns a response indicating that the request is a bad request.

Versions 12 and earlier of the database server had no restriction on the number of HTTP or HTTPS input request variables. Setting the MaxRequestVars protocol option to 0 restores the previous behavior, but doing so is not recommended.

Example

This command starts a database server that accepts requests with up to 100 input variables:

```
dbsrv17 -xs http( MaxRequestVars=100 )
```

Related Information

MaxConnections (MAXCONN) Protocol Option [page 225]
MaxRequestSize (MAXSIZE) Protocol Option [page 227]

1.1.7.27 MyIP (ME) Protocol Option

Use this option for finer control over which network interface controllers (NIC) or adapters to use when several are present on the database server or client computer.

Syntax

```
{ MyIP | ME }={ ip-address [, ip-address ... ] | NONE }
```

Available protocols

TCP/IP, HTTP, HTTPS, OData

Allowed Values

ip-address

This string must be specified in the form of an IP address. You must separate multiple IP addresses with commas.

For OData, you can only specify one IP address.

NONE

If the keyword NONE is supplied as the IP address, no attempt is made to determine local IP addresses. The NONE keyword is intended for clients on computer systems where this operation is expensive, such as computers with multiple network adapters or Remote Access Service (RAS) and a network adapter. It is not intended for use on the database server.

Remarks

For the database server, this option indicates which server IP addresses to use to listen for client connections.

For the client, this option indicates which client IP addresses to try when attempting to communicate with the database server.

Each NIC or network adapter is identified by an IP address. By default, the database server uses every network interface it finds. If you don't want your database server to listen on all network interfaces, specify the address of each interface you want to use in the MyIP (ME) protocol option.

When specifying an IPv6 address on a Windows platform, the interface identifier should be used. Unix platforms support both interface identifiers and interface names in IPv6 addresses. The interface identifier is required on Linux (kernel 2.6.13 and later).

Example

The following command line (entered all on one line) instructs the database server to use two network adapters.

```
dbsrv17 -x tcpip(MyIP=192.75.209.12,192.75.209.32) "%SQLANYSAMP17%\demo.db"
```

The following command line (entered all on one line) instructs the database server to use an IPv6 network adapter:

```
dbsrv17 -x tcpip(MyIP=fe80::5445:5245:444f) "%SQLANYSAMP17%\demo.db"
```

The following connection string fragment instructs the client to make no attempt to determine addressing information.

```
LINKS=tcpip(MyIP=NONE)
```

The following command instructs the database server to listen for OData requests on the specified network adapter:

```
dbsrv17 -xs odata(MyIP=192.75.209.12)
```

Related Information

TCP/IP Protocol [page 176]

IPv6 Support [page 178]

1.1.7.28 QuietConsole (QUIET) Protocol Option

Tells the OData server whether or not to start with its server messages window minimized.

Syntax

```
{ QuietConsole | QUIET }={ YES | NO }
```

Available protocols

OData

Allowed Values

YES Specifying QuietConsole=YES tells the OData server to start with its server messages window minimized. This is the default.

NO Specifying QuietConsole=NO tells the OData server to start with its server messages window displayed.

Default

YES

Remarks

Specify whether or not the OData server starts with its server messages window minimized.

Example

Start a database server that listens on port 8080 for OData requests and starts with the server message window displayed:

```
dbsrv17 -xs odata(ServerPort=8080;QUIET=NO) c:\mydatabase.db
```

1.1.7.29 ReceiveBufferSize (RCVBUFSZ) Protocol Option

Sets the size for the receiving buffer used by the TCP/IP protocol stack.

Syntax

```
{ ReceiveBufferSize | RCVBUFSZ }=size[ k | m | g ]
```

Available protocols

TCP/IP

Allowed Values

size

The value specifies the size of the receive buffer used by the TCP/IP stack. The maximum size that you can specify is 1 MB. This is an advanced option that should only be used if you are familiar with its impact.

Default

Computer-dependent.

Remarks

You may want to increase the value if BLOB performance over the network is important.

Related Information

TCP/IP Protocol [page 176]

1.1.7.30 SecureServerPort Protocol Option

Specifies the port on which the OData server listens for secure (HTTPS) connections.

Syntax

```
SecureServerPort=port-number
```

Available protocols

OData

Allowed Values

port-number

For an OData server, the SecureServerPort protocol option designates the port number on which to communicate using HTTPS.

Specify a single port number, for example (port=1234).

For a client, the SecureServerPort protocol option informs the client of the port or ports on which database servers are listening for HTTPS communications. The client broadcasts to every port that is specified by the SecureServerPort protocol option to find the OData server.

Default

443

Remarks

To disallow HTTPS connections, specify `"SecureServerPort="` or do not specify the SSLKeyStore (KEYSTORE) protocol option.

Example

Run the following command to start a database server that listens on port 8443 for secure OData requests using the key store `c:\mykeys.jks` and the key store password *password*, and specify that the database server does not listen for non-secure OData requests:

```
dbsrv17 -xs odata(SecureServerPort=8443;SSLKeyStore=c:
\mykeys.jks;SSLKeyStorePassword=password;ServerPort=)
```

1.1.7.31 SendBufferSize (SNDBUFSZ) Protocol Option

Sets the size for the sending buffer used by the TCP/IP protocol stack.

Syntax

```
{ SendBufferSize | SNDBUFSZ }=size[ k | m | g ]
```

Available protocols

TCP/IP

Allowed Values

size

The value specifies the size of the send buffer used by the TCP/IP stack. The maximum size that you can specify is 1 MB. This is an advanced option that should only be used if you are familiar with its impact.

Default

Computer-dependent.

Remarks

You may want to increase the value if BLOB performance over the network is important.

Related Information

TCP/IP Protocol [page 176]

1.1.7.32 ServerPort (PORT) Protocol Option

Specifies the port the database server is listening on.

Syntax

```
{ ServerPort | PORT }=port-number
```

Available protocols

TCP/IP, HTTP, HTTPS, OData

Allowed Values

port-number

For a database server, the ServerPort protocol option designates the port number on which to communicate using TCP/IP.

You can specify a single port number, or a combination of individual port numbers and ranges of port numbers. When you specify a list and/or range of port numbers, the database server attempts to bind to all specified port numbers. For example:

- (port=1234)
- (port=1234,1235,1239)
- (port=1234-1238)
- (port=1234-1237,1239,1242)

OData accepts only a single port number. To disallow OData HTTP connections, specify `ServerPort=`. If you are using both HTTP and OData or HTTPS and OData, then one or the other must change port numbers.

For a client, the ServerPort (PORT) protocol option informs the client of the port or ports on which database servers are listening for TCP/IP communication. The client broadcasts to every port that is specified by the ServerPort (PORT) protocol option to find the database server.

Default

TCP/IP

2638
HTTP

80
HTTPS

443
OData 80

TCP/IP remarks

Every application using TCP/IP on a computer uses a distinct TCP/IP **port** so that network packets end up at the right application. The Internet Assigned Numbers Authority has assigned the SQL Anywhere database server port number 2638 to use for TCP/IP communications. However, other applications are not disallowed from using this reserved port, and this may result in an addressing collision between the database server and another application.

On the server

On the server, when you specify the ServerPort (PORT) protocol option, only the specified port number is used for TCP/IP connections. UDP listeners use both the specified port number and the default port number (2638). If you do not specify the ServerPort protocol option, the default port (2638) is used.

On the client

When you specify the ServerPort (PORT) protocol option, only the specified port number is used for TCP/IP connections and UDP broadcasts. If you do not specify the ServerPort protocol option, port 2638 is used.

If the server is running on a port other than 2638, the application can usually connect to the server by providing the server name but no port number. The port number must be specified if the -sb server option is used.

HTTP remarks

If you are using the database server as a web server, it listens on the standard HTTP and HTTPS ports of 80 and 443, respectively, unless otherwise specified.

Example

The following examples show how to use the ServerPort protocol option.

1. Start a network database server:

   ```
   dbsrv17 -x tcpip -n server1
   ```

   Port number 2638 is now taken.
2. Attempt to start another database server:

   ```
   dbsrv17 -x tcpip -n server2
   ```

   The default port is currently allocated, and so the server starts on another port. The port number is displayed on the server messages window.
3. Start a database server on port 2629:

   ```
   dbsrv17 -x tcpip(ServerPort=2629) -n server3 c:\mydata.db
   ```
4. Assuming that server3 was started on host serverhost3, connect to the server3 using the Host connection parameter from dbisql:

   ```
   dbisql -c "UID=DBA;PWD=sql;ServerName=server3;Host=serverhost3:2629"
   ```
5. Assuming that server3 was started on host serverhost3, connect to the server3 using the CommLinks connection parameter from dbisql:

   ```
   dbisql -c
   "UID=DBA;PWD=sql;ServerName=server3;CommLinks=tcpip(HOST=serverhost3;PORT=2629
   )"
   ```

Suppose that another web server on your computer is already using port 80. The following example starts a database server that listens on port 8080 instead:

```
dbsrv17 -xs http(PORT=8080) -n server3 web.db
```

Start a database server that listens on port 8080 for OData requests:

```
dbsrv17 -xs odata(ServerPort=8080) c:\mydatabase.db
```

Related Information

-x Database Server Option [page 523]
-xs Database Server Option [page 529]
-sb Database Server Option [page 490]

1.1.7.33 skip_certificate_name_check Protocol Option (Client Side Only)

Controls whether the client library skips the check of the server host name against the database server certificate host names.

Available protocols

HTTPS, TLS

Default

None

Remarks

The shorter form *skip_cert_name_check* is also accepted.

Set this boolean option to ON or OFF to control whether the host name of the database server matches any of the host names in the root certificate. Enabling this option may prevent the client from fully authenticating the server, leaving it vulnerable to attack.

When initiating TLS or HTTPS connections, the client libraries check the host name of the database server against the certificate provided by that server. This check only occurs if none of the certificate_name, certificate_company, or certificate_unit options are specified, or the skip_certificate_name_check option is not enabled. If any of certificate_name, certificate_company, or certificate_unit are specified, then only those options are verified. The skip_certificate_name_check option disables the host name check when enabled.

The host names or IP addresses are derived from the subjectAltName (Subject Alternative Name or SAN) extension and the Common Name (CN) field. The SAN may contain multiple host names with wild cards. For example, a Google certificate might include *.google.com, *.google.ca, and *.android.com. Thus, www.google.ca is a valid host name.

HTTPS is only supported for web services client procedures.

Example

The following connection string fragment verifies that `myrootcert.crt` is at the root of the database server's certificate's signing chain, but does not verify that myhost.com is identified in the certificate.

```
HOST=myhost.com;SERVER=myserver;ENCRYPTION=TLS(trusted_certificate=myrootcert.crt
;skip_certificate_name_check=ON)
```

Related Information

trusted_certificate Protocol Option [page 244]
Encryption (ENC) Connection Parameter [page 80]

1.1.7.34 SSLKeyStore (KEYSTORE) Protocol Option

Specifies the file path for the Java Key Store containing the SSL certificate that the OData server uses to encrypt communication.

Syntax

```
{ SSLKeyStore | KEYSTORE }=filepath
```

Available protocols

OData

Allowed Values

filepath The file path for the Java Key Store (JKS format).

Default

None

Remarks

This option must be specified to enable secure OData requests.

The Java key store must contain only the one public/private key pair certificate that the OData server uses to encrypt communication. The Java key store should contain the public keys of intermediary signing authorities and may contain the public key of the root signing authority in the signing chain. The key store's password must be the same as the OData server certificate's password.

The file used as the key store contains important security information. It should be in a directory with restricted access.

Example

Run the following command to start a database server that listens on port 8443 for secure OData requests using the key store `c:\mykeys.jks` and the key store password *password*, and specify that the database server does not listen for non-secure OData requests:

```
dbsrv17 -xs odata(SecureServerPort=8443;SSLKeyStore=c:
\mykeys.jks;SSLKeyStorePassword=password;ServerPort=)
```

Related Information

SSLKeyStorePassword (KEYSTOREPASSWORD) Protocol Option [page 240]

1.1.7.35 SSLKeyStorePassword (KEYSTOREPASSWORD) Protocol Option

Specifies the password used by the OData server to authenticate against the Java Key Store identified by the SSLKeyStore (KEYSTORE) protocol option.

Syntax

```
{ SSLKeyStorePassword | KEYSTOREPASSWORD }=password
```

Available protocols

OData

Allowed Values

password The password to authenticate against the Java Key Store and the server certificate.

Default

None

Remarks

The key store's password must be the same as the OData server certificate's password.

Example

Run the following command to start a database server that listens on port 8443 for secure OData requests using the key store `c:\mykeys.jks` and the key store password *password*, and specify that the database server does not listen for non-secure OData requests:

```
dbsrv17 -xs odata(SecureServerPort=8443;SSLKeyStore=c:
\mykeys.jks;SSLKeyStorePassword=password;ServerPort=)
```

Related Information

SSLKeyStore (KEYSTORE) Protocol Option [page 239]

1.1.7.36 TDS Protocol Option (Server Side Only)

Controls whether Tabular Data Stream (TDS) connections to a database server are allowed.

Syntax

TDS={ *YES* | *NO* | *ENCRYPTED*}

Available protocols

TCP/IP (server side only)

Default

YES

Remarks

Set the TDS connection parameter to either:

Value	Description
YES (default)	Allow unencrypted TDS connections, and encrypted ones as well if the -ec TLS setting is supplied. See -ec Database Server Option [page 418].
NO	Block all encrypted and unencrypted TDS connection requests.
ENCRYPTED	Allow encrypted TDS connections (configured by the -ec "TLS" option), but reject unencrypted TDS requests. See -ec Database Server Option [page 418].

Example

The following command starts a database server using the TCP/IP protocol, but disallows connections from SAP Open Client, jConnect, or other TDS-based applications.

```
dbsrv17 -x tcpip( TDS=NO ) ...
```

Related Information

-ec Database Server Option [page 418]

1.1.7.37 Timeout (TO) Protocol Option

Specifies the length of time, in seconds, to wait for a response when establishing communications (TCP/IP), and the length of time, in seconds, to wait for a request (HTTP or HTTPS).

Syntax

```
{ Timeout | TO }=timeout-value
```

Available protocols

TCP/IP, HTTP, HTTPS

Allowed Values

timeout-value

- When using TCP/IP, this integer specifies the length of time, in seconds, to wait for a response when establishing communications. The maximum length of time that you can specify is 3600 seconds.
- When using HTTP or HTTPS, this integer specifies the length of time, in seconds, to wait to receive a complete request.

Default

TCP/IP

5

HTTP

30

HTTPS

30

Remarks

On the client, this protocol option specifies how long to wait for:

- The UDP broadcast response when searching for the server.
- The TCP/IP connect call to make the connection to the database server.
- The response from the database server when disconnecting.

When using TCP/IP on the server, this protocol option specifies how long to wait for:

- The UDP broadcast response when searching for mirroring or diagnostic servers to make connections to.
- The TCP/IP connect call when making mirroring or diagnostic connections to other database servers.
- The response when disconnecting from mirroring or diagnostic servers.

When using HTTP or HTTPS on the database server, this protocol specifies:

- The maximum idle time permitted when receiving a request. The Timeout period begins when the request starts and ends when the request finishes. If this limit is reached, the connection is closed and a request timeout message is returned to the client.

Example

The following connection string fragment tells the client libraries to use TCP/IP to connect to the database server, with a timeout of 20 seconds:

```
CommLinks=tcpip(TO=20)
```

Related Information

KeepaliveTimeout (KTO) Protocol Option [page 213]

1.1.7.38 trusted_certificate Protocol Option

Specifies the path and file name of a file that contains one or more trusted certificates or a PEM-encoded string.

Syntax

```
trusted_certificate={ public-certificate-filename | PEM-encoded-certificate | * | none }
```

Available protocols

TLS, HTTPS

Default

None

Remarks

The keyword *trusted_certificates* is also accepted.

The trusted_certificate encryption protocol option is used to specify the trusted certificate or its location.

The trusted_certificate encryption protocol option value is either the name of a file that contains a collection of PEM-encoded X.509 trusted root certificates, the PEM-encoded certificates directly, "*" to search for the trusted certificate in the operating system certificate store, or *NONE*.

If TLS is specified in the Encryption connection parameter, then this protocol option should be used. If this protocol option is omitted, the operating system certificate store will be searched by default.

Clients use the trusted_certificate option to ensure that they are connecting to the correct database server. Similarly, database servers use the trusted_certificate option to ensure that clients can only connect if they are using a certificate signed by one that the database server trusts. The trusted certificate can be a server's self-signed certificate or a certificate belonging to a commercial Certificate Authority. You must generate your certificates using RSA encryption.

Specify *NONE* for the trusted_certificate protocol option value to make a TLS connection without verifying the server's certificate. No trusted root certificate is required on the client. Connecting without verifying the server's certificate is less secure than verifying the certificate because the client can no longer protect against a man-in-the-middle attack. However, the connection is still strongly encrypted and prevents replay attacks, which is more secure than no encryption at all.

When initiating TLS or HTTPS connections, the database client libraries check the host name of the database server against the certificate provided by that server using the procedure described in RFC 2818. This check is done for additional security.

The host names or IP addresses are derived from the Common Name (CN) field and subjectAltName (Subject Alternative Name or SAN) extension. The SAN may contain multiple host names with wild cards. For example, a Google certificate might include *.google.com, *.google.ca, and *.android.com. Thus, www.google.ca is a valid host name.

Suppose that the host name of the server is myserver.sample.com. Then the Common Name (CN) field in the server's identity certificate is checked to see if it matches this host name. If it does not, then the subjectAltName (SAN) is checked. If the SAN contains *.sample.com, then a suitable match has been found. If there is no suitable match, a TLS handshake failure will occur.

The host name check is performed if none of the certificate_name, certificate_company, or certificate_unit options are specified, or the skip_certificate_name_check option is not enabled.

If any of certificate_name, certificate_company, or certificate_unit are specified, then only those options are verified. In these cases, the setting of the skip_certificate_name_check option is ignored.

If specified, the certificate_name must match the Common Name field of the server certificate. The Common Name (CN) is normally the fully qualified domain name of the server using the certificate.

If specified, the certificate_company must match the Organization field of the server certificate. The Organization (O) is typically the name of the company.

If specified, the certificate_unit must match the Organizational Unit field of the server certificate. The Organizational Unit (OU) is typically the name of the department within the company.

In the absence of one of these three options, skip_certificate_name_check=yes can be used to skip the host name check.

Enabling the skip_certificate_name_check option is not recommended since it may prevent the client from fully authenticating the server, leaving it vulnerable to attack.

HTTPS is only supported for web services client procedures.

Example

The following example specifies the certificate file and certificate name.

```
ENCRYPTION=TLS(trusted_certificate=myrootcert.crt;certificate_name=host.sample.co
m)
```

In this example, the client library:

- verifies that `myrootcert.crt` is at the root of the server's certificate's signing chain,
- skips the default host name check, and
- verifies that the common name field of the certificate is set to host.sample.com.

The following example specifies the certificate file only.

```
ENCRYPTION=TLS(trusted_certificate=myrootcert.crt)
```

In this example, the client library:

- verifies that `myrootcert.crt` is at the root of the server's certificate's signing chain, and
- verifies that one of the host names or IP addresses in the server's certificate matches the host name or IP address that we are connecting to.

The following example specifies the certificate file and requests that the certificate name check be skipped.

```
ENCRYPTION=TLS(trusted_certificate=myrootcert.crt;skip_certificate_name_check=tru
e)
```

In this example, the client library:

- verifies that `myrootcert.crt` is at the root of the server's certificate's signing chain,
- skips the default host name check, and
- does no other checking.

The following example specifies that the client's trusted certificate is in the operating system certificate store and that the server's certificate Common Name is SAPNetCA. The default host name check is skipped.

```
ENCRYPTION=TLS(trusted_certificate=*;certificate_name=SAPNetCA)
```

The following example connects Interactive SQL to the database server named demo by using transport layer security. The default host name check is skipped.

```
dbisql -c "UID=DBA;PWD=sql;Server=demo;Host=localhost;
    ENCRYPTION=TLS(fips=no;
       trusted_certificate=rsaroot.crt;
       certificate_name=RSA Server;
       certificate_company=SAP;
       certificate_unit=SQL Anywhere)"
```

In the following example, the trusted certificate is provided directly in the connection string:

```
dbisql -c
"UID=DBA;PWD=sql;Server=demo;Host=localhost;ENCRYPTION=TLS(fips=no;trusted_certif
icate='-----BEGIN CERTIFICATE-----
MIIDkDCCAnigAwIBAgICAQEwDQYJKoZIhvcNAQELBQAweDELMAkGA1UEBhMCQ0ExCzAJBgNVBAgMAk9OM
REwDwYDVQQHDAhXYXRlcmxvbzEMMAoGA1UECgwDU0FQMRUw…etc.etc.KFsmOQ==-----END
CERTIFICATE----- '
```

```
       certificate_name=RSA Server;certificate_company=SAP;certificate_unit=SQL
Anywhere)"
```

Related Information

SQL Anywhere Client/Server Communication Encryption [page 1745]
Separately Licensed Components
Using Client-side Certificates for TLS [page 1744]
certificate_company Protocol Option (Client Side Only) [page 195]
certificate_name Protocol Option (Client Side Only) [page 196]
certificate_unit Protocol Option (Client Side Only) [page 198]
Encryption (ENC) Connection Parameter [page 80]
Certificate Creation Utility (createcert) [page 1130]
CREATE PROCEDURE Statement [Web Service]
skip_certificate_name_check Protocol Option (Client Side Only) [page 238]
identity Protocol Option [page 210]
identity_password Protocol Option [page 211]

1.1.7.39 VerifyServerName (VERIFY) Protocol Option (Client Side Only)

Controls whether clients must verify the database server name before connecting.

Syntax

```
{ VerifyServerName | VERIFY }={ YES | NO }
```

Available protocols

TCP/IP (client side only)

Default

YES

Remarks

If the client uses the CommLinks (LINKS) connection parameter for a TCP/IP connection and the client succeeds in making the connection, it verifies that the name of the server found is the same as the one it is looking for. If no server name was specified using the ServerName (Server) connection parameter or if the server name does not match, the connection will fail. Specifying VerifyServerName=NO skips the verification of the server name. This allows SQL Anywhere clients to connect to a SQL Anywhere server if they know only an IP address/port.

> **i** Note
>
> It is recommended that you use the Host connection parameter instead of the CommLinks (LINKS) connection parameter. When the Host connection parameter is used and both the host name and port are specified, it is not required that you specify a server name. However, it is recommended that a server name always be specified to guarantee that a connection is made to the correct server.

Related Information

CommLinks (LINKS) Connection Parameter [page 59]
Host Connection Parameter [page 86]
ServerName (Server) Connection Parameter [page 110]

1.1.8 Connection Limits for Databases and Database Servers

When you make a connection to a database, you connect via the database server. The maximum number of concurrent connections to your database depends on several factors.

These factors include:

- Your license.
- The type of database server that you choose to run.
- The number of databases running on the database server.
- The type of connections that the database and database server accept.
- The database and database server options that you specify to reduce the maximum number of connections for your application.

Database Server Connection Limits

The connection limit for the personal database server (dbeng17) is ten connections, and by default only seven of these connections can be HTTP connections. At minimum three connections are reserved for standard connections. A standard connection is any connection other than an HTTP or HTTPS connection

The connection limit for network database servers is equal to the maximum number of connections allowed by your per-seat license, or unlimited (32766) connections if you have a per-core license. (In practice, creating 32766 connections to a single database server might not be possible because of operating system restrictions. Also the database server makes internal temporary connections during operation, which also reduces the number of connections available for users.)

To reduce the maximum number of concurrent connections that a database server can accept, specify the -gm database server option.

If your database server accepts HTTP/HTTPS connections, then you can limit the maximum number of HTTP/HTTPS connections that can be made to a database server by specifying the MaxConnections protocol option. You can also reserve database connections for only standard connections by setting the reserved_connections database option.

Database Connections

If your database server has numerous databases running on it, then limiting the number of connections to a specific database can be more effective than limiting connections to the database server. For example, a database server that allows 100 connections may have 90 connections to one database, leaving only ten connections available for all other databases.

To limit the number of connections that a database can accept, set the maximum_connections database option for that database.

If your database accepts HTTP/HTTPS connections and you must ensure that standard connections can be accepted at any time, then reserve connections for standard connections. Specify the reserved_connections database option to reserve standard connections.

You can also use a login policy to limit the number of connections to users who have the specific login policy. When creating the login policy, set the max_connections login policy option or the max_non_dba_connections login policy options. Alternatively, you could create a custom login procedure to limit the number of connections by using the login_procedure database option.

HTTP/HTTPS Connection Queuing and Limiting Connections

The database server queues HTTP/HTTPS connections. When the database server reaches its connection limit and a new HTTP/HTTPS connection is attempted, the connection attempt is queued. Connection attempts remain in the queue until an HTTP/HTTPS connection becomes available, or until the attempt times out after 150 seconds. The queue can hold a maximum of 1000 connection attempts.

Standard connections are not queued. When a database server reaches its connection limit, any new standard connection attempt fails.

While there are connection attempts in the HTTP/HTTPS queue, there is no opportunity for a standard connection to replace an HTTP/HTTPS connection. A user wanting to make a standard connection could wait indefinitely unless some of connections are reserved for standard connections. By reserving some of the database's connections for standard connections, you allow users to connect to the database to perform administrative tasks, such as dropping connections, performing backups, and so on.

Reserve standard connections by using the reserved_connections database option. The network database server does not reserve connections for standard connections. The personal database server reserves a minimum of 3 connections for standard connections.

You could also limit the number of HTTP/HTTPS connections that can be made to a database server by specifying the MaxConnections protocol option.

Connections Above the Connection Limit

When the connection limit has been reached, a DBA user can make one additional standard connection to a database above the limit to forcibly drop connections as needed. This new connection is allowed only if *all* of the following conditions are met:

- The user has the DROP CONNECTION or SERVER OPERATOR system privilege.
- The total number of database server connections (excluding the new connection) is less than or equal to the MaxConnections database server property limit.
- The total number of database connections (excluding the new connection) is less than or equal to the MaxConnections database property limit.
- The connection is a standard connection.

In this section:

Related Information

Connection Listeners [page 253]
max_connections Option [page 761]
SET OPTION Statement
RESERVED_CONNECTIONS Option [page 811]
MaxConnections (MAXCONN) Protocol Option [page 225]
-gm Database Server Option [page 436]
List of Database Server Properties [page 900]
List of Database Properties [page 917]
login_procedure Option [page 755]
Login Policy Options and Default Values [page 642]

1.1.8.1 Limiting Database Connections

Limit the number of concurrent connections to a database by using the max_connections database option..

Prerequisites

You must have the SET ANY SYSTEM OPTION system privilege.

Context

If your database server has numerous databases running on it, then limiting the number of connections to a specific database can be more effective than limiting connections to the database server. For example, a database server that allows 100 connections may have 90 connections to one database, leaving only ten connections available for all other databases.

If the max_connections database option is not set, then the database server connection limit is applied to the database.

Procedure

Choose one of the following options:

Option	Action
Set the max_connections database option permanently.	Execute a SET OPTION statement to specify the number of concurrent connections that are allowed to the database. For example: `SET OPTION PUBLIC.max_connections = 25;`
Set the max_connections database option temporarily.	Execute a SET TEMPORARY OPTION state to temporarily limit the number of concurrent connections allowed by the database. The specified limit is in place for as long as the database is running. For example: `SET TEMPORARY OPTION PUBLIC.max_connections = 25;`

Results

Concurrent connections to the database or database server are limited to the number specified.

Related Information

Connection, Database, and Database Server Properties [page 879]
max_connections Option [page 761]
SET OPTION Statement

1.1.8.2 Reserving Standard Connections

Specify the minimum number of connections that can accept standard connections.

Prerequisites

You must have the SET ANY SYSTEM OPTION system privilege.

Context

When the connection limit for HTTP/HTTPS has been reached, new HTTP/HTTPS connections are queued until a connection becomes available. While there are HTTP/HTTPS connections in the queue, a standard connection cannot succeed unless some of the database connections are reserved for standard connections.

By reserving some of the database's connections for standard connections, you allow users to connect to the database to perform administrative tasks, such as dropping connections, performing backups, and so on.

By default, the personal database server reserves a minimum of 3 connections for standard connections.

The network database server does not reserve any connections.

Procedure

Choose one of the following options:

Option	Action
Set the reserved_connections database option permanently	Execute a SET OPTION statement to specify the number of concurrent connections that must be reserved for standard connections. This option is only applicable when the database supports HTTP/HTTPS connections. For example: `SET OPTION PUBLIC.reserved_connections = 25;`

Option	Action
Set the reserved_connections database option temporarily	Execute a SET TEMPORARY OPTION state to temporarily specify the number of concurrent connections that must be reserved for standard connections. This option is only applicable when the database supports HTTP/HTTPS connections. The specified number is in place for as long as the database is running. For example: `SET TEMPORARY OPTION PUBLIC.reserved_connections = 25;`

Related Information

Connection, Database, and Database Server Properties [page 879]
RESERVED_CONNECTIONS Option [page 811]
SET OPTION Statement

1.1.9 Connection Listeners

Connection listeners allow connections to a database server using a specific network protocol, address, or port.

A connection listener listens on a particular address for connection attempts from specified network protocols. The database server can have listeners for shared memory, HTTP, HTTPS, and TCP/IP. All connection listeners except for shared memory listen on an IP address and a specific port. When you start a database server, use the -x or -xs database server option to tell the database server which type of connection attempts or web requests to listen for.

If the IP addresses change on the computer where the database server is running, then the database server responds by automatically starting or stopping connection listeners as necessary. For example, if a database server is running on a laptop and the laptop then connects to a new wireless network, the database server detects the change and starts a connection listener on the new IP address. Or, if the laptop disconnects from a network, then the database server detects the disconnection and shuts down the appropriate connection listeners.

Connection listeners can also be manually started and stopped without stopping and restarting the database server by running the sp_start_listener and sp_stop_listener system procedures, or by using the *Listeners* tab for the database server in SQL Central.

In this section:

Creating a Connection Listener (SQL Central) [page 254]
Create connection listeners to allow connections to a database server using a different network protocol, address, or port from those specified when the database server started. You can create connection listeners for shared memory, TCP/IP, HTTP, or HTTPS.

Deleting a Connection Listener (SQL Central) [page 255]
Deleting connection listeners prevents new connections from being accepted on the specified network protocol, address, or port. Existing connections are not affected.

Related Information

sp_start_listener System Procedure
sp_stop_listener System Procedure

1.1.9.1 Creating a Connection Listener (SQL Central)

Create connection listeners to allow connections to a database server using a different network protocol, address, or port from those specified when the database server started. You can create connection listeners for shared memory, TCP/IP, HTTP, or HTTPS.

Prerequisites

You must have EXECUTE privilege on the sp_start_listener system procedure and the MANAGE LISTENERS system privilege.

Context

TCP/IP connection listeners use the encryption setting specified with the -ec database option when the database server started.

You can create shared memory connection listeners regardless of the -ec database server option specified when the database server starts. Shared memory connection listeners started with sp_start_listener allow unencrypted connections to the database server.

Procedure

1. In SQL Central, use the SQL Anywhere 17 plug-in to connect to the database.
2. In the left pane, select the server.
3. In the right pane, click the *Listeners* tab
4. Right-click and click *New* > *Listener* and follow the instructions in the wizard.

Results

The connection listener is created.

Related Information

sp_start_listener System Procedure
sp_stop_listener System Procedure
sp_start_listener System Procedure
sp_stop_listener System Procedure

1.1.9.2 Deleting a Connection Listener (SQL Central)

Deleting connection listeners prevents new connections from being accepted on the specified network protocol, address, or port. Existing connections are not affected.

Prerequisites

You must have EXECUTE privilege on the sp_stop_listener system procedure and the MANAGE LISTENERS system privilege.

Context

Procedure

1. In SQL Central, use the SQL Anywhere 17 plug-in to connect to the database.
2. In the left pane, select the server.
3. In the right pane, click the listener that you want to stop.
4. Right-click and click *Delete*

Results

The connection listener is deleted.

Related Information

sp_start_listener System Procedure

sp_stop_listener System Procedure
sp_start_listener System Procedure
sp_stop_listener System Procedure

1.1.10 Connection IDs

When a user connects to a database, the database server assigns the user's connection a unique **connection ID**.

A database server connection forms a channel through which all activity from the client application takes place. Client applications cannot interact with the database server until a connection is made. When a database server connection is made, a user's privileges determine what actions the user is authorized to perform on the database server.

Connection IDs for temporary connections start at one billion.

For each new connection to the database server, the server increments the connection ID value by one. These connection IDs are logged in the database server message log, which records informational messages, errors, warnings, and messages from the MESSAGE statement. The connection ID can be used to perform the following tasks:

- filter request logging information
- identify which connection has a lock on the database
- track the total number of connections to a database server since it started and the order in which those connections were made

You can use the CONNECTION_PROPERTY function to obtain a user's connection ID by querying the Number connection property.

Related Information

Advanced Topic: Temporary Connections [page 258]
Database Server Logging [page 336]
Request Logging [page 1502]
How Locking Works
-z Database Server Option [page 532]

1.1.11 Improve Application Performance with Connection Pooling

Connection pooling may improve the performance of applications that make multiple, brief connections to the database server.

If connection pooling is enabled for a connection, when it is disconnected, the connection is automatically cached and may be reused when the application reconnects. For server-side connection pooling, enable

pooling with the ConnectionPool (CPOOL) connection parameter. Once an application makes five connections with the same connection string, then the connection is pooled.

For connection pooling in .NET applications, use the .NET *Pooling* connection parameter - the ConnectionPool (CPOOL) connection parameter is ignored. The .NET Data Provider cleans up and reinitializes pooled connections.

Connection pooling is not supported for Tabular Data Stream (TDS) connections.

An application must make five connections with the same connection string before a connection is cached. The connection name can be different each time using the ConnectionName (CON) connection parameter, but all other connection parameter values must be identical for a cached connection to be reused. The following two connection strings are considered equivalent for connection pooling.

```
UID=DBA;PWD=passwd;ConnectionName=One
Password=passwd;UserID=DBA;CON=Two
```

If the application process connects again and there are cached connections available for the same connection string, the cached connection is reused. Connections remain in the cached state for the time specified by the ConnectionPool (CPOOL) connection parameter (60 seconds by default).

This does not apply to .NET connection pooling. Multiple connection pools are supported by .NET and a different connection string creates a different pool. To reuse a pooled connection, the connection strings must be textually identical. For .NET applications, the following two connection strings are not considered equivalent for connection pooling; thus each string would create its own connection pool.

```
UID=DBA;PWD=passwd;ConnectionName=One
UserID=DBA;PWD=passwd;ConnectionName=One
```

Connection pooling does not occur for non-standard database authentication such as Integrated or Kerberos logins. Only user ID and password authentication is supported.

Cached connections are not reused if it would change the behavior of the application. For example, cached connections are not reused:

- For databases that stop automatically when there are no connections to them.
- If connections are disabled.
- If the database server has reached its connection limit.
- If a password has changed.
- If the login_mode option is set.

To ensure that connection pooling is transparent to the application, a connection is disconnected if a failure occurs when caching a connection. If a failure occurs when attempting to reuse a cached connection, the database server attempts to connect normally.

A connection is cached if it is disconnected and the maximum number of connections specified by the CPOOL connection parameter has not been reached. The connection is reinitialized, and the cached connection continues to be connected to the database server even though the application has disconnected it. The cleanup and reinitialization of a connection includes the following activities:

- Rolling back all outstanding transactions.
- Dropping temporary tables, temporary functions, and variables.
- Resetting connection options and connection counters.

- Decrementing and incrementing the database server connection counts. You are not informed that there are active connections when a database server with only cached connections shuts down.
- Executing all defined disconnect and connect events.
- Executing the login_procedure database option and verifying the login policy.
- Resetting the connection ID.

Using SQL Anywhere Connection Pooling with Other Connection Pooling Products

If you are using a product or API that supports connection pooling, then the connection pooling of the product or API supersedes SQL Anywhere connection pooling. Both types of connection pooling can be active at the same time.

The behavior of connection pooling in your product or the API may be significantly different than SQL Anywhere connection pooling. If the behavior of connection pooling in your product or API is inappropriate for an application, SQL Anywhere connection pooling can be used and may improve the performance of some applications.

Connection Pooling and Read-Only Scale-Out

If the NodeType (NODE) connection parameter is also specified for a connection, the application typically connects to the primary server and the primary server determines which copy node is least heavily loaded. The connection is then redirected to that node. If the application makes and drops several such connections within a short period of time, the connection is pooled and the primary server is not asked which copy node to use. This behavior reduces the load on the primary server, but may not give expected behavior.

Related Information

Connection Parameters and Connection Strings [page 35]
.NET Connection Pooling
ConnectionPool (CPOOL) Connection Parameter [page 66]

1.1.12 Advanced Topic: Temporary Connections

The database server uses temporary connections to perform operations such as running backups or initializing databases.

You can get information about temporary connections by using the sa_conn_info or sa_conn_list system procedure. The ParentConnection property returns the connection ID of the connection that spawned the temporary connection.

Temporary connections have connection IDs that are larger than 1 billion (1000000000), and their names describe the function of the connection.

The following example uses the sa_conn_info system procedure to return a result set showing which connection created a temporary connection.

```
SELECT Number, Name, ParentConnection FROM sa_conn_info();
```

Connection 8 spawned the temporary connection that executed a CREATE DATABASE statement.

```
Number        Name               ParentConnection
-------------------------------------------------------------
1000000048    INT: CreateDB      8
9             SQL_DBC_14675af8   (NULL)
8             SQL_DBA_152d5ac0   (NULL)
```

Related Information

sa_conn_info System Procedure
sa_conn_list System Procedure
CONNECTION_PROPERTY Function [System]

1.1.13 Troubleshooting: Connections

An understanding of how connections are established can help you resolve connectivity problems.

Ensure you are familiar with communication protocols and network-specific issues, including connections across firewalls.

To establish a connection, the software:

1. Locates the interface library.
2. Assembles a list of connection parameters.
3. Locates a database server. If the database server is not found, then a personal server is started.
4. Locates the database.

The connection procedure is the same for:

- Any **ODBC application** using the SQLDriverConnect function, which is the common connection method for ODBC applications. Many application development systems, such as SAP PowerBuilder, belong to this class of application. The SQLConnect function is also available to ODBC applications.
- Any client application using **Embedded SQL** and using the recommended function for connecting to a database (db_string_connect). In addition, the CONNECT SQL statement is available for Embedded SQL applications and in Interactive SQL. It has two forms: CONNECT AS and CONNECT USING. All the database utilities, including dbisqlc, use db_string_connect.
- Any **.NET** application using ADO.NET. The application creates a new SAConnection object and passes the connection string to the constructor or sets the ConnectionString property. Then the application calls the Open method on the SAConnection object to connect.

- Any **ADO application** using the ADODB Connection object. The Provider property is used to locate the OLE DB driver. The ConnectionString property may use DataSource as an alternative to DataSourceName and User ID as an alternative to UserID.
- Any application using the **SQL Anywhere JDBC driver** to pass the URL `jdbc:sqlanywhere:` followed by a standard connection string as a parameter to the Driver Manager.GetConnection method.

In this section:

Related Information

Communication Protocols [page 173]

Troubleshooting: Database Server Startup [page 1023]
Troubleshooting Network Communications [page 1028]

1.1.13.1 Troubleshooting: The Location of the Interface Library

Generally, the location of this DLL or shared library is transparent to the user.

ODBC Driver Location

For ODBC, the interface library is also called an ODBC driver. An ODBC client application calls the ODBC driver manager, and the driver manager locates the SQL Anywhere driver.

The ODBC driver manager searches the supplied data source to locate the driver. When you create a data source using the ODBC Data Source Administrator or dbdsn utility, the utility fills in the current location for your ODBC driver. The data source information is stored in the Windows Registry, or in the Unix system information file (named `.odbc.ini` by default).

Embedded SQL Interface Library Location

Embedded SQL applications are normally linked against the Embedded SQL import library. The interface library is loaded by name when the application starts. The name of the Embedded SQL interface library is:

Windows

`dblib17.dll`

Unix

`libdblib17` with an operating-system-specific extension

OLE DB Driver Location

The provider name (SAOLEDB) is used to locate the OLE DB Provider DLL (`dboledb17.dll`) based on entries in the registry. The entries are created when the SAOLEDB provider is installed or if it is re-registered.

ADO.NET

ADO.NET programs add a reference to the SQL Anywhere ADO.NET provider, which is named `Sap.Data.SQLAnywhere.dll`. The .NET Data Provider DLL is added to the .NET Global Assembly Cache (GAC) when it is installed.

JDBC Driver Location

When you run your application, the Java package `sajdbc4.jar` must be in the classpath. The system must be able to locate the native DLLs or shared objects.

Windows

On Windows, the current directory, the system path, and the `Windows` and `Windows\system32` directories are searched.

Unix

On Unix, the system path and the user library path are searched.

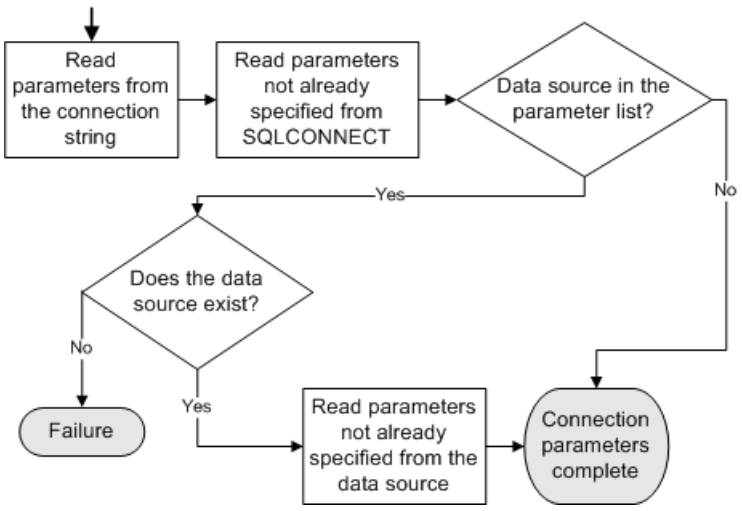
When the Library is Located

A connection string is sent to the interface library when it is located by the client application. The string is used by the interface library to assemble a list of connection parameters, and establish a server connection.

1.1.13.2 Troubleshooting: How Connections Are Established

There is a sequence of actions that take place when a connection is attempted.

The following diagram illustrates how the interface library assembles the list of connection parameters and establishes a connection.



Precedence

Parameters held in more than one place are subject to the following order of precedence:

1. Connection string
2. SQLCONNECT environment variable
3. ODBC data source

> **Note**
>
> If a connection string and an ODBC data source both specify the same connection parameter, the value from the connection string is used and the value from the data source is ignored.

Failure

Failure at this stage can occur if the connection string uses incorrect syntax, an unknown connection parameter, or if, in the connection string or in SQLCONNECT, you specify a data source that does not exist.

Ignored parameters

Depending on other connections already in use, some connection parameters may be ignored, such as:

AutoStop

Ignored if the database is already loaded.

DatabaseFile

Ignored if DatabaseName is specified and a database with this name is already running.

The interface library uses the completed list of connection parameters to attempt to connect.

Related Information

Connection Parameter Syntax Rules [page 40]

1.1.13.3 Troubleshooting: How Database Servers Are Located

When SQL Anywhere locates a running server, it tries to locate or start the required database on that server.

If SQL Anywhere cannot locate a running server, it may attempt to start a personal server, depending on the connection parameters.

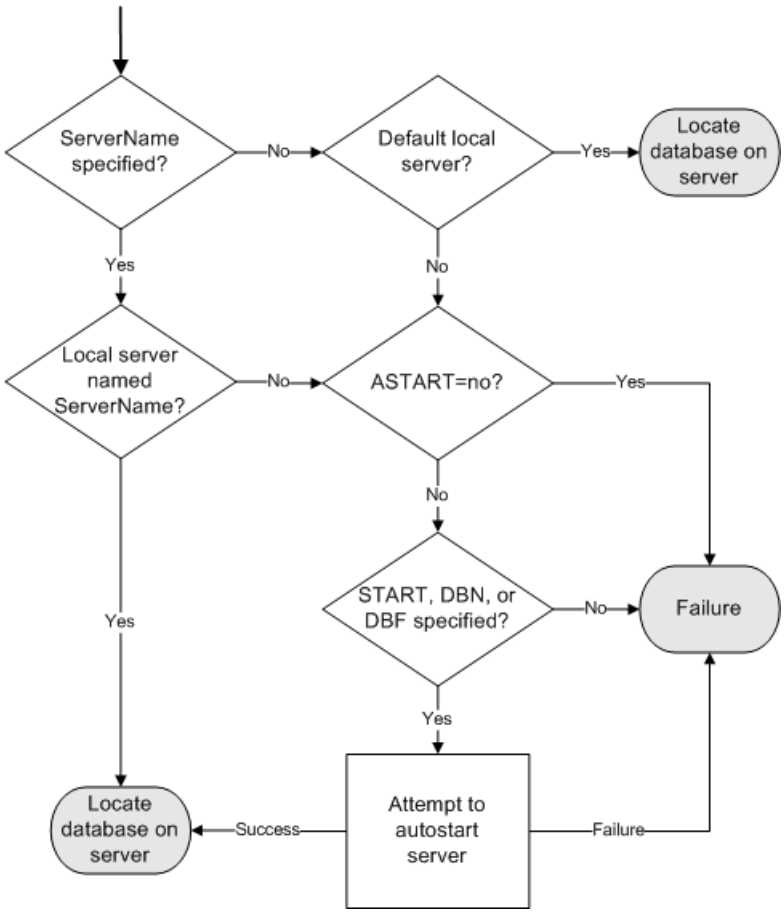
> **Note**
>
> - For local connections, locating a server is simple. For connections over a network, the Host connection parameter is recommended.
> - If the server is started automatically, then information from the START, DBF, DBKEY, DBS, DBN, Server, and AutoStop connection parameters is used to construct the options for the automatically started server. The server does not automatically start if the Host or the CommLinks=TCPIP connection parameters are specified.

- If the server has an alternate server name, you can only use the alternate server name to connect to the database that specified the alternate server name. You cannot use the alternate server name to connect to any other databases running on that database server.

The following diagrams illustrate how SQL Anywhere locates a database server.

Connection String Does Not Include the CommLinks or the Host Connection Parameters

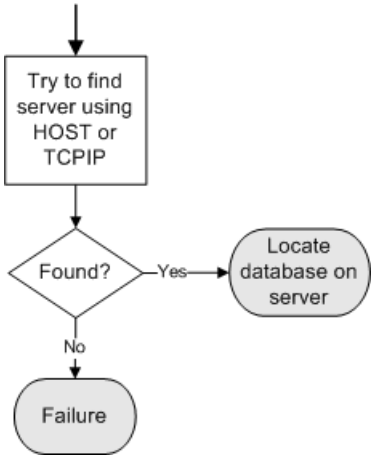
The following diagram describes how SQL Anywhere locates a database server using shared memory when the connection string does not include the CommLinks or the Host connection parameters.



If SQL Anywhere cannot locate a running server, it may attempt to start a personal server, depending on the connection parameters.

Connection String Includes the Host Connection Parameter or the CommLinks Connection Parameter Specifies TCPIP

The following diagram describes how SQL Anywhere locates a database server when the connection string includes either the Host connection parameter or the CommLinks connection parameter, and it specifies TCPIP.



Connection String Includes a CommLinks Connection Parameter that Is Set to ShMem and TCPIP

When the connection string includes either CommLinks=ShMem, TCPIP or CommLinks=TCPIP, ShMem, then SQL Anywhere:

- Attempts a shared memory connection, excluding attempts to autostart the database server.
  If SQL Anywhere fails to locate the server using shared memory, then it attempts a TCPIP connection.

Related Information

Troubleshooting: How Database Servers Are Located [page 564]
Troubleshooting: How the Host Connection Parameter Locates Database Servers [page 267]
-xd Database Server Option [page 526]
Server Enumeration Utility (dblocate) [page 1200]
db_string_ping_server Function
Troubleshooting: How the Host Connection Parameter Locates Database Servers [page 267]
Troubleshooting: How the CommLinks=TCPIP Connection Parameter Locates Database Servers [page 270]

1.1.13.4 Troubleshooting: How the Broadcast Repeater Utility Locates Database Servers

Find database servers running on other subnets and through firewalls where UDP broadcasts normally do not reach, without using the Host connection parameter or LDAP.

Context

The Broadcast Repeater utility (dbns17) allows clients to find database servers running on other subnets and through firewalls where UDP broadcasts normally do not reach, without using the Host connection parameter or LDAP.

Any number of DBNS processes can communicate with each other. Each DBNS process connects to every other DBNS that it is aware of, and the different DBNS processes share their lists of DBNS processes. For example, suppose you start two DBNS processes, A and B. If you start a third DBNS process, C, in a third subnet, passing the address of B to C, then B tells C about A, and C then connects to A.

> **i** Note
>
> Running more than one DBNS process in a single subnet is generally not necessary, and is not recommended.
>
> If you must run multiple DBNS processes in a single subnet, then you must specify a different port for each DBNS process. A single DBNS process can only listen on one port. Use the -ap parameter in the Broadcast Repeater utility (dbns17).

If either the Host connection parameter or the HOST protocol option is used, then the Broadcast Repeater utility is not needed.

Procedure

1. Start a DBNS (database name service) process on any computer in a subnet.
2. Start a DBNS process on any computer in a different subnet and pass the computer name or IP address of the first computer as a parameter.

   The two DBNS processes make a TCP/IP connection to each other.
3. The DBNS processes now listen for broadcasts on each of their own subnets. Each DBNS process forwards requests over the TCP/IP connection to the other DBNS process, which re-broadcasts the requests on its subnets and also forwards responses back to the originating DBNS process, which sends them to the original client.

Results

Regular broadcasts on either of the subnets reach database servers on the remote subnet, and clients are able to connect to database servers on the remote subnet without specifying the HOST parameter.

Related Information

Broadcast Repeater Utility (dbns17) [page 1128]

1.1.13.5 Troubleshooting: How the Host Connection Parameter Locates Database Servers

When attempting to find a database server over TCP/IP using the Host connection parameter, the client tries several steps to find the server.

Once the client has successfully connected to the database server, it attempts to connect to the database. If the database server is not found, then an error specifying that the server is not found is returned. If a different error occurs, then the connection attempt fails without attempting further steps.

These are the steps the client performs until it finds a server:

1. check the database server address cache (`sasrv.ini`)
2. attempt a TCP/IP connection to the address(es) specified in the Host connection parameter
3. send out UDP find database server requests

Step 1: Check the Database Server Address Cache (`sasrv.ini`)

The client checks its `sasrv.ini` file for an entry that matches the database server name specified by the ServerName connection parameter. If the connection string includes the Host connection parameter but not ServerName, the client cannot use the `sasrv.ini` file.

If no matching cache entry is found, the client attempts a direct TCP/IP connection.

If a cache entry matches both the ServerName connection parameter and the address specified by the Host connection parameter, the client attempts a TCP/IP connection to the IP address listed in the cache.

- If the client connects to the database server using this IP address, it compares the database server's name to the value specified by ServerName.
  If the database server name matches, the connection is successful.
  If the database server name does not match, the client removes the entry from its `sasrv.ini` file, and the client attempts a direct TCP/IP connection.
- If the client cannot connect, it removes the entry from its `sasrv.ini` file and attempts a direct TCP/IP connection.

For example, assume there is a computer with the host name kangaroo and the IP address 10.25.13.5. A database server named joey is running on this computer on port 49152.

For the connection string `Host=kangaroo;ServerName=joey`, the client finds an entry in its `sasrv.ini` file that matches the server name joey. The IP address in the cache is 10.25.13.5:49152. The client compares the address of the specified Host name with 10.25.13.5. The addresses match, so the client connects over TCP/IP to 10.25.13.5:49152 and then verifies that the connected server is named joey. The connection is successful.

Step 2: Attempt a TCP/IP Connection to the Address(es) Specified in the Host Connection Parameter

When multiple addresses are specified, the client attempts to connect to each address in the order that they are specified in the Host connection parameter. The client attempts connections until it connects successfully or it has tried all the addresses.

When an address includes a port, the client attempts a TCP/IP connection to the address and port number specified. If no port is specified, then the client attempts a TCP/IP connection at the specified address on the default port, 2638.

- If the client connects to the database server and ServerName is specified, the client compares the database server's name to the value specified by ServerName.
  If the database server name matches, then the connection is successful. The client updates its `sasrv.ini` file.
  If the database server name does not match, then the client attempts to connect using UDP find server requests.
- If the client connects to the database server and ServerName is not specified, then the connection is successful.
- If the client cannot connect and a ServerName is specified, then the client attempts to connect using UDP find server requests.
- If the client cannot connect and ServerName is not specified, then the connection fails.

For example, assume there is a computer with the host name kangaroo and the IP address 10.25.13.5. A database server named joey is running on this computer on port 49152. This is the first time that the client has connected to this server, so there is no cached address in its `sasrv.ini` file.

For the connection string `Host=kangaroo:49152;ServerName=joey`, the client connects over TCP/IP to 10.25.13.5:49152 and then verifies that the connected server is named joey. The connection is successful, so the client updates its `sasrv.ini` file.

Step 3: Send out UDP Find Database Server Requests

The SQL Anywhere client only sends out UDP find database server requests when the connection string includes the ServerName connection parameter and the Host value does not specify a port. UDP find server requests are sent to port 2638 on the specified addresses.

By default, all the database servers on the same computer listen for UDP packets on port 2638. When a database server receives a UDP find server request, the database server compares its name to the name

specified in the UDP request. If the names match, then the database server sends back a UDP response packet containing its IP address and port number. Only the matching database server sends a response packet.

By default, the client waits up to five seconds for a UDP response. If no responses are received, the UDP packets are re-sent every second until this process times out.

When the client receives the UDP response packet, the client attempts a TCP/IP connection to the address and port specified in the packet.

If the client can connect to a database server, then the connection is successful. The client updates its `sasrv.ini` file.

> **i** Note
>
> UDP packets cannot find the following database servers:
>
> - A database server that has the -sb 0 option specified.
> - A database server whose UDP requests are blocked by a firewall, router, or gateway.
>
> In each of these cases, specifying the port of the database server in the Host connection parameter should allow the client to connect successfully.

For example, assume there is a computer with the host name kangaroo and the IP address 10.25.13.5. A database server named joey is running on this computer on port 49152. This is the first time that the client has connected to this server, so there is no cached address in its `sasrv.ini` file.

For the connection string `Host=kangaroo;ServerName=joey`, the client attempted to connect over TCP/IP to 10.25.13.5:2638 and it failed (Step 2). Because the connection string does not have a port number specified, the client sends UDP find server requests to address 10.25.13.5:2638. The database server joey is listening on UDP address 10.25.13.5.2638 and responds with the address 10.25.13.5:49152. The client connects over TCP/IP to 10.25.13.5:49152. The connection is successful, so the client updates its `sasrv.ini` file.

Related Information

Firewall Connections [page 179]

Database server not found

Troubleshooting: How Database Servers Are Located [page 564]

Troubleshooting: How Database Server Address Information Is Cached in sasrv.ini for Faster Connections [page 273]

-sb Database Server Option [page 490]

ServerPort (PORT) Protocol Option [page 235]

1.1.13.6 Troubleshooting: How the CommLinks=TCPIP Connection Parameter Locates Database Servers

A SQL Anywhere client attempting to find a server over TCP/IP using the CommLinks connection parameter performs several steps to find the server.

If the database server is not found, then an error specifying that the server is not found is returned. If a different error occurs, then the connection attempt fails without attempting further steps. Once the client has successfully connected to the database server, it attempts to connect to the database.

It is recommended that you only use the CommLinks (LINKS) connection parameter if you need to specify TCP/IP protocol options other than HOST or ServerPort (PORT). Otherwise, use the Host connection parameter.

These are the steps the client performs:

Step 1: Check the Database Server Address Cache (`sasrv.ini`)

If the connection string includes the HOST protocol option but not the ServerName connection parameter, the LocalOnly protocol option is set to YES, or the DoBroadcast protocol option is set to NONE, then the client attempts a direct TCP/IP connection. That is, the client does not use the database server address cache or the LDAP server.

The client checks its `sasrv.ini` file for an entry that matches the database server name specified in the ServerName connection parameter. If the connection string also includes the HOST protocol option, then the cache entry must also match the host address specified.

If no matching cache entry is found, the client attempts to find the database server using LDAP.

When a matching cache entry is found, the client attempts a TCP/IP connection to the IP address listed in the cache.

- If the client connects to the database server using this IP address and the VerifyServer protocol option is set to No, the connection is successful.
- If the client connects to the database server using this IP address and VerifyServer is set to Yes (the default), the client compares the database server's name to the value specified by ServerName. If the database server name matches, the connection is successful. If the database server name does not match, the client removes the entry from its `sasrv.ini` file, and the client attempts to find the database server using LDAP.
- If the client cannot connect, it removes the entry from its `sasrv.ini` file and attempts to find the database server using LDAP.

For example, assume there is a computer with the host name kangaroo and the IP address 10.25.13.5. A database server named joey is running on this computer on port 49152.

For the connection string `CommLinks=TCPIP(Host=kangaroo);ServerName=joey`, the client finds an entry in its `sasrv.ini` file that matches the server name joey. The IP address in the cache is 10.25.13.5:49152. The client compares the address of the specified Host name with 10.25.13.5. The addresses match, so the client connects over TCP/IP to 10.25.13.5:49152 and then verifies that the connected server is named joey. The connection is successful.

Step 2: Check the LDAP Server for the Server Name

The client queries the LDAP server for an entry that matches the ServerName connection parameter when the SQL Anywhere client and database server computers are configured to use an LDAP server.

Step 3: Attempt a TCP/IP Connection to the Address(es) Specified in the HOST Protocol Option

If the HOST protocol option is not specified, the client attempts to find the Server using UDP find server requests.

The client attempts to connect over TCP/IP to the addresses specified in the HOST protocol option. When multiple addresses are specified, the client attempts to connect to each address in the order that they are specified in the HOST protocol option. The client attempts connections until it connects successfully or it has tried all the addresses.

When the connection string includes a port (in either the HOST protocol option or the ServerPort connection property), the client attempts a TCP/IP connection to the address and port number specified. If no port is specified, then the client attempts a TCP/IP connection at the specified address on the default port, 2638.

- If the client connects to the database server using the specified address and the VerifyServer protocol option is set to No, the connection is successful.
- If the client connects to the database server using the specified address, the client compares the database server's name to the value specified by ServerName. If the database server name matches, the connection is successful and the client updates its `sasrv.ini` file. If the database server name does not match, the client attempts to connect using UDP find server requests.
- If the client cannot connect and ServerName is specified, the client attempts to connect using UDP find server requests.

For example, assume there is a computer with the host name kangaroo and the IP address 10.25.13.5. A database server named joey is running on this computer on port 49152. This is the first time that the client has connected to this server, so there is no cached address in its `sasrv.ini` file.

For the connection string `CommLinks=TCPIP(Host=kangaroo:49152);ServerName=joey`, the client connects over TCP/IP to 10.25.13.5:49152 and then verifies that the connected server is named joey. The connection is successful, so the client updates its `sasrv.ini` file.

Step 4: Send out UDP Find Database Server Requests

The SQL Anywhere client only sends out UDP find database server requests when the connection string includes the ServerName connection parameter, and the DoBroadcast protocol option is set to DIRECT or ALL.

- When DoBroadcast=DIRECT or the HOST protocol option is specified, the client sends UDP find server packets to the address(es) specified. When a port is specified (in either the HOST protocol option or the ServerPort protocol option), the UDP find server request is sent to the specified port; otherwise, the request is sent to the default port, 2638.

- When DoBroadcast=ALL and no HOST protocol option is specified, the client determines the subnet's broadcast address for each of its IP addresses. If the ServerPort protocol option is specified, the client sends UDP broadcast find server packets to the specified port; otherwise, the UDP broadcast packets are sent to the default port, 2638.

Unless the -sb 0 option has been specified when the database server started, then all the database servers listen for UDP packets on port 2638. When a database server receives a UDP find server request, the database server compares its name to the name specified in the request packet. If the names match, then the database server sends back a UDP response packet containing its IP address and port number. Only the matching database server sends a response packet.

By default, the client waits up to five seconds for a UDP response. If no responses are received, the UDP packets are re-sent every second until this process times out.

When the client receives the UDP response packet, the client attempts a TCP/IP connection to the address and port specified in the packet.

If the client can connect to a database server, then the connection is successful. The client updates its `sasrv.ini` file.

> **i Note**
>
> UDP packets cannot find the following database servers:
>
> - A database server that is located on a different subnet. UDP find server broadcast packets cannot find the database server unless the Broadcast Repeater Utility is used.
> - A database server that has the -sb 0 option specified.
> - A database server whose UDP requests are blocked by a firewall, router, or gateway.
>
> In each of these cases, specifying the address (including the port number) of the database server in the HOST protocol option should allow the client to connect successfully.

For example, assume there is a computer with the host name kangaroo and the IP address 10.25.13.5. A database server named joey is running on this computer on port 49152. This is the first time that the client has connected to this server, so there is no cached address in its `sasrv.ini` file. The connection is being attempted from a computer on the same subnet as kangaroo, and the broadcast address for this subnet is 10.25.13.255.

For the connection string `CommLinks=TCPIP;ServerName=joey`, the client sends UDP find server broadcast requests to the broadcast address 10.25.13.255:2638. The database server joey is listening on UDP address 10.25.13.5.2638 and responds with the address 10.25.13.5:49152. The client connects over TCP/IP to 10.25.13.5:49152. The connection is successful, so the client updates its `sasrv.ini` file.

Related Information

Connections Using LDAP as a Name Server [page 181]
Firewall Connections [page 179]
Host Connection Parameter [page 86]
Database server not found
Troubleshooting: How Database Servers Are Located [page 564]

Troubleshooting: How Database Server Address Information Is Cached in sasrv.ini for Faster Connections [page 273]
VerifyServerName (VERIFY) Protocol Option (Client Side Only) [page 247]
Broadcast Repeater Utility (dbns17) [page 1128]
ServerPort (PORT) Protocol Option [page 235]
-sb Database Server Option [page 490]

1.1.13.7 Troubleshooting: How Database Server Address Information Is Cached in `sasrv.ini` for Faster Connections

When using the Host or CommLinks connection parameter, attempting multiple TCP/IP connections or broadcasting over large networks to search for a database server of a specific name can be time-consuming.

Caching database server addresses speeds up network connections by saving the database server name and IP address to the `sasrv.ini` file to be used for subsequent connections.

The cache file contains a set of sections, each of the following form:

```
[Server name]
Link=TCPIP
Address=ip_address:port
```

> **i** Note
>
> It is important that each database server has a unique name. Giving different database servers the same name can lead to identification problems.

Location of the `sasrv.ini` File

The default location of the `sasrv.ini` file is:

Operating system	Default location of `sasrv.ini`
Windows	`%APPDATA%\SQL Anywhere 17`
Unix	`$HOME/.sqlanywhere17`

Related Information

Troubleshooting: How the Host Connection Parameter Locates Database Servers [page 267]
Troubleshooting: How the CommLinks=TCPIP Connection Parameter Locates Database Servers [page 270]

1.1.13.8 Troubleshooting: How to Test Connection Strings (dbping)

Use the dbping utility to troubleshoot connections.

You can use the dbping utility with the -c option to test connection string parameters. By default, dbping does not connect to a database, it only connects to the server and it does not start the server. Use the -d option with the dbping utility to connect to the database and start the database server, if necessary.

Example

For example, suppose a personal server with the name demo17 is running the sample database (which can be started with the command `dbeng17 -n demo17 "%SQLANYSAMP17%\demo.db"`). The following command returns a message indicating that the ping was successful if a database server named demo17 is running on the local computer and has a database named demo running:

```
dbping -d -c "Server=demo17;DBN=demo;UID=DBA;PWD=sql"
```

The following command tests to see if a database server named demo17 is running with the database named demo and starts the server and database if it is not running:

```
dbping -d -c "Server=demo17;DBN=demo;DBF=c:\sa17\samples
\demo.db;ASTOP=no;UID=DBA;PWD=sql"
```

The following command tests to see if a database server can be found on a computer named Waterloo on the default port 2638:

```
dbping -c "Host=Waterloo"
```

The following command tests to see if a default database server is running on the current computer.

```
dbping
```

Related Information

Ping Utility (dbping) [page 1195]
Troubleshooting: How to Test Embedded SQL and Network Connection Performance (dbping) [page 275]

1.1.13.9 Troubleshooting: How to Test Embedded SQL and Network Connection Performance (dbping)

Use the Ping utility (dbping) to obtain information about the performance of Embedded SQL connections and the network by specifying the -s or -st options.

The following statistics are gathered:

Statistic	Description
DBLib connect and disconnect	The time to perform one DBLIB connect and disconnect. The performance of connecting and disconnecting using other interfaces, such as ODBC, is typically slower than DBLIB because more requests are required to complete the connection.
Round trip simple request	The time it takes to send a request from the client to the server plus the time it takes to send a response from the server back to the client. The round trip time is twice the average latency.
Send throughput	The throughput based on transferring 100 KB of data for each iteration from dbping to the database server.
Receive throughput	The throughput based on transferring 100 KB of data for each iteration from the database server to dbping.

If your network has both high round trip times and high throughput, then reported throughput will be lower than your actual network throughput because of the high round trip times. Using dbping -s can be useful to give an indication of whether communication compression may improve performance. The performance statistics are approximate, and are more accurate when both the client and server computers are fairly idle. The transferred data can be compressed to approximately 25% of its original size if communication compression is used:

The following is an example of the output from dbping -s for the dbping command:

```
dbping -s -c "UID=DBA;PWD=sql;Server=demo17;DBN=demo"
```

```
SQL Anywhere Server Ping Utility Version 17.0.11.1293
Connected to SQL Anywhere 17.0.11.1293 server "demo17" and database "demo".
Performance Statistic           Number            Total Time    Average
----------------------------    --------------    ----------    ------------
DBLib connect and disconnect      200 times         512 msec         2 msec
Round trip simple request       22100 requests     1024 msec        <1 msec
Send throughput                267500 KB           1024 msec    261230 KB/sec
Receive throughput             230400 KB           1024 msec    225000 KB/sec
Ping database successful.
```

Related Information

Ping Utility (dbping) [page 1195]
Troubleshooting: How to Test Connection Strings (dbping) [page 274]

1.1.13.10 Troubleshooting: Compatible Protocol Options for Client and Database Servers

Ensure that the client and database server are using the same protocol.

Shared Memory Protocol

If the client and database server are on the same computer, shared memory (the default protocol) is recommended.

TCP/IP protocol

If the client and database server are on different computers, you must use TCP/IP. The network server (dbsrv17) allows TCP/IP connections by default.

Clients use the Host or CommLinks connection parameters to establish TCP/IP connections to the database server.

The personal server (dbeng17) does not allow TCP/IP connections by default. The personal server can accept local (but not remote) TCP/IP connections by specifying the -x tcpip option.

Related Information

-x Database Server Option [page 523]
Host Connection Parameter [page 86]
CommLinks (LINKS) Connection Parameter [page 59]

1.1.13.11 Troubleshooting: Common Connection Problems and Solutions

If you receive an error message indicating that the database server cannot be found when trying to connect, the client cannot find the database server on the network.

Check for the following problems:

- Specify the server name in the connection string.
  Since a computer system can run multiple database servers, always specify the database server name (ServerName=`server-name`) when connecting to a database.
- For remote database servers, it is recommended that you specify the Host connection parameter and provide the TCP/IP host name or address of the computer running the database server. Verify that the specified address is correct. If the server is using a port other than the default 2638, you may need to specify Host=`tcpip-address:port-number`.
  If you are using the advanced CommLinks connection parameter, packets used to find the database server may not be getting to the database server. It is recommended that you use the Host connection parameter unless you need to use TCP/IP protocol options other than Host, ServerPort, or DoBroadcast. If you are using the CommLinks connection parameter without specifying the HOST protocol option, UDP broadcasts or requests may be limited to the current subnet or be blocked by a router, gateway, or firewall.

- A firewall between the client and server may be preventing the connection.
- The personal server only accepts connections from the same computer. If the client and server are on different computers, you must use the network server.
- Your network drivers are not installed properly or the network wiring is not installed properly. Use the ping utility to verify that the client computer can communicate with the server computer.
- The server must use the TCP/IP protocol if you are connecting via SAP Open Client or jConnect.

Related Information

Firewall Connections [page 179]
Network Protocol Options [page 187]
Troubleshooting: Connections [page 259]
Troubleshooting: Database Server Startup [page 1023]
Database server not found
Host Connection Parameter [page 86]

1.2 Database Creation

To create a database, you define the tables it will have (entities), the columns in each table (attributes), and the relationships between tables (keys and constraints).

You can use SQL Central, the initialization utility (dbinit), and the CREATE DATABASE statement to create or **initialize** a database. After creating the database, you can connect to it and add tables and other objects.

Transaction Log

When you create a database, you must decide where to place the transaction log. This log stores all changes made to a database, in the order in which they are made. In the event of a media failure on a database file, the transaction log is essential for database recovery. It also makes your work more efficient. By default, it is placed in the same directory as the database file, but this configuration is not recommended for production use.

Database File Compatibility

Database files are compatible among all operating systems, except where file system file size limitations or support for large files apply.

A database created on any operating system can be used on another operating system by copying the database file(s). Similarly, a database created with a personal database server (dbeng17) can be used with a

network database server (dbsrv17). Database servers can manage databases created with earlier versions of the software, but old servers cannot manage newer databases..

In this section:

Related Information

Tables, Views, and Indexes
The Transaction Log [page 292]
Physical Limitations on Size and Number of Databases [page 936]
Viewing Entity-relationship (ER) Diagrams [page 1108]

1.2.1 Creating a Database (SQL Central)

Create a database using SQL Central.

Prerequisites

By default, you must have the SERVER OPERATOR system privilege. The required privileges can be changed by using the -gu database server option.

Procedure

1. In SQL Central, click *Tools* > *SQL Anywhere 17* > *Create Database*.
2. Follow the instructions in the *Create Database Wizard*.

Results

A database is created.

Related Information

Connection IDs [page 256]
Data Import
Creating a Database (dbinit Utility) [page 279]
Creating a Table

1.2.2 Creating a Database (dbinit Utility)

Use the Initialization utility (dbinit) to create a database.

Context

With this utility, you can include command line options to specify different settings for the database.

A transaction log mirror provides extra protection for critical data and enables complete data recovery if a media failure on the transaction log occurs. To maintain a transaction log mirror when you create a database, specify the -m option.

Procedure

Run a dbinit command.

Results

A database is created.

Example

For example, to create a database called `company.db` with a 4 KB page size, run the following command:

```
dbinit -dba DBA,passwd -p 4k company.db
```

The following command (which should be entered on one line) initializes a database named `company.db`, with a transaction log kept on a different device and a mirror on a third device.

```
dbinit -dba DBA,passwd -t d:\log-dir\company.log -m
e:\mirr-dir\company.mlg c:\db-dir\company.db
```

Related Information

Connection IDs [page 256]
Transaction Log Mirrors [page 301]
Initialization Utility (dbinit) [page 1166]
CREATE DATABASE Statement

1.2.3 Database File Types

Each database has several files associated with it.

The database file

This file holds the entire contents of your database. A single file can contain a single database, or you can add up to 12 dbspaces, which are additional files holding portions of the same database. You choose a location for the database file and dbspaces. The database file typically has the extension `.db`.

The transaction log

This file holds a record of the changes made to the database, and is necessary for synchronization and recovery of the information in your database in the event of a failure. The transaction log typically has the extension `.log`.

The rollback log (also called the undo log)

As changes are made to the contents of a database, a rollback log is created so that changes can be canceled if a transaction is rolled back or if a transaction is uncommitted when a system failure occurs. There is a separate rollback log for each connection. When a transaction is committed or rolled back, the contents of the rollback log for that connection are deleted. The rollback logs are stored in the database, and rollback log pages are copied into the checkpoint log along with other pages that are changed.

The temporary file

The database server uses the temporary file during a database session when the database server needs more space than is available to it in the cache for such operations as sorting and forming unions. When the database server needs this space, it generally uses it intensively. The overall performance of your database becomes heavily dependent on the speed of the device containing the temporary file. The database server discards this file once the database shuts down even if the database server remains running. The temporary file has a server-generated name with the extension `.tmp`.

The location of the temporary file can be specified when starting the database server using the -dt server option. If you do not specify the location of the temporary file when starting the database server, the following environment variables are checked, in order:

- SATMP environment variable
- TMP environment variable
- TMPDIR environment variable
- TEMP environment variable

If none of these environment variables are defined, the database server places its temporary file in the current directory on Microsoft Windows operating systems, or in the `/tmp` directory on UNIX and Linux.

The database server creates, maintains, and removes the temporary file. You only need to ensure that there is enough free space available for the temporary file. You can obtain information about the space available for the temporary file using the sa_disk_free_space procedure.

Predefined dbspace files

These files store your data and other files used by the database.

Dbspace files

You can spread your data over several separate files, in addition to the database file.

Transaction log mirror files

For additional security, you can create a mirror copy of the transaction log. This file typically has the extension `.mlg`.

In this section:

Dropping a Database (dberase Utility) [page 314]
Drop a database by using the dberase utility.

Related Information

Transaction Log Mirrors [page 301]
sa_disk_free_space System Procedure

1.2.3.1 Predefined Dbspaces

The database server uses predefined dbspaces for its databases.

Dbspace	Name
Main database file	system
Temporary file	temporary or temp
Transaction log file	translog
Transaction log mirror	translogmirror

You cannot create user-defined dbspaces with these names and you cannot drop the predefined dbspaces.

If you upgrade a version 10.0.0 or earlier database with user-defined dbspaces that use the predefined dbspace names, then all references to these dbspaces in SQL statements are assumed to be referring to the user-defined dbspaces, and not the predefined dbspaces. The only way that you can refer to the predefined dbspaces is by dropping the user-defined dbspaces, or renaming them to not use the same names as the predefined dbspaces.

The ALTER DBSPACE statement supports the predefined dbspace names so you can add more space to them.

The DB_EXTENDED_PROPERTY function also accepts the predefined dbspace names.

Related Information

ALTER DBSPACE Statement
DB_EXTENDED_PROPERTY Function [System]

1.2.3.2 Additional Dbspaces Considerations

Additional database files allow you to cluster related information in separate files.

> **i** Note
>
> For most databases, a single database file is enough. However, for users of large databases, additional database files are sometimes necessary.

When you initialize a database, it contains one database file. This first database file is called the **main file** or the **system** dbspace. By default, all database objects and all data are placed in the main file.

A **dbspace** is an additional database file that creates more space for data. A database can be held in up to 13 separate files (the main file and 12 dbspaces). Each table, together with its indexes, must be contained in a single database file. The SQL statement CREATE DBSPACE adds a new file to the database.

Temporary tables are only created in the temporary dbspace.

There are several ways to specify the dbspace where a base table or other database object is created. In the following lists, the location specified by methods occurring earlier in the list take precedence over those occurring later in the list.

1. IN DBSPACE clause (if specified)
2. default_dbspace option (if set)
3. system dbspace

If a dbspace name contains a period and is not quoted, the database server generates an error for the name.

Each database file has a maximum allowable size of 2^{28} (approximately 268 million) database pages. For example, a database file created with a database page size of 4 KB can grow to a maximum size of one terabyte (2^{28}*4 KB). However, in practice, the maximum file size allowed by the physical file system in which the file is created affects the maximum allowable size significantly.

While some older file systems restrict file size to a maximum of 2 GB, many file systems, such as Windows using the NTFS file system, allow you to exploit the full database file size. In scenarios where the amount of data placed in the database exceeds the maximum file size, it is necessary to divide the data into more than one database file. As well, you may want to create multiple dbspaces for reasons other than size limitations, for example, to cluster related objects.

You can use the sa_disk_free system procedure to obtain information about space available for a dbspace.

The SYSDBSPACE system view contains information about all the dbspaces for a database.

Splitting existing databases

To split existing database objects among multiple dbspaces, you must unload your database and modify the generated script file (named `reload.sql` by default) for rebuilding the database. In the `reload.sql` file, add IN clauses to the CREATE TABLE statements to specify the dbspace for each table you do not want to place in the main file.

In this section:

Privileges on Dbspaces [page 284]
Only the CREATE privilege is supported on dbspaces.

Dbspace Creation [page 285]
You create a new database file, or dbspace, either from SQL Central, or using the CREATE DBSPACE statement.

Preallocating Database File Space (SQL Central) [page 288]
Preallocate disk space for dbspaces or transaction logs to improve performance when loading large amounts of data.

Preallocating Database File Space (SQL) [page 289]
Preallocate disk space for dbspaces or transaction logs to improve performance when loading large amounts of data.

Dropping a Dbspace (SQL Central) [page 290]
Drop a dbspace in SQL Central.

Related Information

Physical Limitations on Size and Number of Databases [page 936]
sa_disk_free_space System Procedure
SYSDBSPACE System View

1.2.3.2.1 Privileges on Dbspaces

Only the CREATE privilege is supported on dbspaces.

The CREATE privilege allows a user to create database objects in the specified dbspace. You can grant CREATE privilege for a dbspace by executing a GRANT CREATE ON statement.

The CREATE privilege on dbspaces behaves as follows:

- A user trying to create a new object with underlying data must have CREATE privilege on the dbspace where the data is being placed.
- Even if a GRANT CREATE ON statement was issued, the user (grantee) must have the CREATE ANY OBJECT system privilege to create new database objects.
- The current list of objects that can be placed in specific dbspaces, and that require the CREATE privilege, includes tables, indexes, text indexes, and materialized views. Objects such as normal views and procedures do not have any underlying data and do not require the CREATE privilege.
- A user can be granted the CREATE privilege directly, or they can inherit the privilege through membership in a role that has been granted the privilege.
- It is possible to grant PUBLIC the CREATE privilege on a specific dbspace, in which case any user who also has the CREATE ANY OBJECT system privilege can create objects on the dbspace.
- A newly created dbspace automatically grants CREATE privilege on itself to PUBLIC.
- It is possible to revoke privileges, for example when trying to secure a dbspace. Privileges on the internal dbspaces system and temporary can also be managed to control access.

- Creating local temporary tables does not require any privileges; dbspace privileges do not affect the creation of local temporary tables. However, the creation of global temporary tables requires the CREATE ANY OBJECT system privilege and CREATE privilege on the temporary dbspace.

Related Information

GRANT Statement
CREATE DBSPACE Statement
DB_EXTENDED_PROPERTY Function [System]
CREATE TABLE Statement
UNLOAD Statement

1.2.3.2.2 Dbspace Creation

You create a new database file, or dbspace, either from SQL Central, or using the CREATE DBSPACE statement.

The database file for a new dbspace can be located on the same disk drive as the main file or on another disk drive.

You must have the MANAGE ANY DBSPACE system privilege to create dbspaces.

For each database, you can create up to twelve dbspaces in addition to the main dbspace. A newly created dbspace is empty. When you create a new table or index you can place it in a specific dbspace with an IN clause in the CREATE statement or set the default_dbspace option before creating the table. If you don't specify an IN clause, and don't change the setting of the default_dbspace option, the table is created in the system dbspace.

Each table is contained entirely in the dbspace it is created in. By default, indexes appear in the same dbspace as their table, but you can place them in a separate dbspace by supplying an IN clause as part of the CREATE statement.

In this section:

Creating a Dbspace (SQL Central) [page 286]
Create a dbspace using SQL Central.

Creating a Dbspace (SQL) [page 287]
Create a dbspace using the CREATE DBSPACE statement.

Related Information

default_dbspace Option [page 723]
CREATE DBSPACE Statement
CREATE TABLE Statement
CREATE INDEX Statement

1.2.3.2.2.1 Creating a Dbspace (SQL Central)

Create a dbspace using SQL Central.

Prerequisites

You must have the MANAGE ANY DBSPACE system privilege.

Context

A single database file is enough for most databases, but additional dbspaces can be used to create more space for data.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *Dbspaces*.
3. Right-click the folder and click *New* > *Dbspace*.
4. Follow the instructions in the *Create Dbspace Wizard*.

Results

The new dbspace appears in the *Dbspaces* folder.

Related Information

Tables
CREATE DBSPACE Statement
default_dbspace Option [page 723]
CREATE INDEX Statement

1.2.3.2.2.2 Creating a Dbspace (SQL)

Create a dbspace using the CREATE DBSPACE statement.

Prerequisites

You must have the MANAGE ANY DBSPACE system privilege.

Context

A single database file is enough for most databases, but additional dbspaces can be used to create more space for data.

Procedure

Execute a CREATE DBSPACE statement.

Results

The dbspace appears in the same directory as the main database file, unless otherwise specified.

Example

The following command creates a new dbspace called MyLibrary in the file `library.db` in the same directory as the main file:

```
CREATE DBSPACE MyLibrary
AS 'library.db';
```

The following command creates a table LibraryBooks and places it in the MyLibrary dbspace.

```
CREATE TABLE LibraryBooks (
title CHAR(100),
author CHAR(50),
isbn CHAR(30)
) IN MyLibrary;
```

The following commands create a new dbspace named MyLibrary, set the default dbspace to the MyLibrary dbspace, and then create the LibraryBooks table in the MyLibrary dbspace.

```

```

Related Information

Tables
CREATE DBSPACE Statement
default_dbspace Option [page 723]
CREATE INDEX Statement

1.2.3.2.3 Preallocating Database File Space (SQL Central)

Preallocate disk space for dbspaces or transaction logs to improve performance when loading large amounts of data.

Prerequisites

You must have the MANAGE ANY DBSPACE system privilege.

Context

When you create a new database file, you can preallocate database file space. Preallocating space can improve performance for loading large amounts of data and keeps database files more contiguous.

As you use a database, it automatically grows database files as needed. Rapidly changing database files can lead to excessive file fragmentation on the disk, resulting in potential performance problems. As well, many small allocations are slower than one large allocation. If you are working with a database with a high rate of change, you can preallocate disk space for dbspaces or for transaction logs.

> → Tip
>
> Running a disk defragmentation utility after preallocating disk space helps ensure that the database file is not fragmented over many disjointed areas of the disk drive. Performance can suffer if there is excessive fragmentation of database files.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *Dbspaces*.
3. In the right pane, right-click the dbspace and click *Preallocate Space*.
4. Enter the amount of space to add to the dbspace. You can add space in units of pages, bytes, kilobytes (KB), megabytes (MB), gigabytes (GB), or terabytes (TB).

5. Click *OK*.

Results

Database file space is preallocated.

Related Information

Dbspace Creation [page 285]
ALTER DBSPACE Statement
CREATE DATABASE Statement
Initialization Utility (dbinit) [page 1166]

1.2.3.2.4 Preallocating Database File Space (SQL)

Preallocate disk space for dbspaces or transaction logs to improve performance when loading large amounts of data.

Prerequisites

You must have the MANAGE ANY DBSPACE system privilege.

Context

When you create a new database file, you can preallocate database space using the DATABASE SIZE clause of the CREATE DATABASE statement. Preallocating space keeps database files more contiguous.

As you use a database, it automatically grows database files as needed. Rapidly changing database files can lead to excessive file fragmentation on the disk, resulting in potential performance problems. As well, many small allocations are slower than one large allocation. If you are working with a database with a high rate of change, you can preallocate disk space for dbspaces or for transaction logs using either SQL Central or the ALTER DBSPACE statement.

> → Tip
>
> Running a disk defragmentation utility after preallocating disk space helps ensure that the database file is not fragmented over many disjointed areas of the disk drive. Performance can suffer if there is excessive fragmentation of database files.

Procedure

1. Connect to a database.
2. Execute an ALTER DBSPACE statement.

Results

Database file space is preallocated.

Example

Increase the size of the system dbspace by 200 pages.

```
ALTER DBSPACE system
ADD 200;
```

Increase the size of the system dbspace by 400 megabytes.

```
ALTER DBSPACE system
ADD 400 MB;
```

Related Information

Dbspace Creation [page 285]
ALTER DBSPACE Statement
CREATE DATABASE Statement
Initialization Utility (dbinit) [page 1166]

1.2.3.2.5 Dropping a Dbspace (SQL Central)

Drop a dbspace in SQL Central.

Prerequisites

You must have the MANAGE ANY DBSPACE system privilege.

Before you can delete a dbspace, you must delete all tables and indexes that use the dbspace.

Procedure

1. Use the *SQL Anywhere 17* plug-in to connect to the database.
2. Open the *Dbspaces* folder.
3. Right-click the dbspace and click *Delete*.

Results

The dbspace is deleted.

Related Information

DROP DBSPACE Statement
DROP TABLE Statement
DROP INDEX Statement

1.2.3.3 Dropping a Dbspace (SQL)

Drop a dbspace using the DROP DBSPACE statement in SQL.

Prerequisites

You must have the MANAGE ANY DBSPACE system privilege.

Before you can drop a dbspace, you must drop all tables and indexes that use the dbspace.

Procedure

1. Connect to a database.
2. Execute a DROP DBSPACE statement.

Results

The dbspace is dropped.

Related Information

DROP DBSPACE Statement
DROP TABLE Statement
DROP INDEX Statement

1.2.3.4 The Transaction Log

The **transaction log** is a file that stores all changes to the database. For example, inserts, updates, deletes, commits, rollbacks, and database schema changes are all logged.

The transaction log is also called the **forward log** or the **redo log**.

The transaction log is a key component of backup and recovery, and is also essential for data synchronization using MobiLink or for data replication using SQL Remote.

By default, all databases use transaction logs. Using a transaction log is optional, but you should use a transaction log unless you have a specific reason not to. Running a database with a transaction log provides greater protection against failure, better performance, and the ability to replicate data.

It is recommended that you store the database files and the transaction log on separate disks on the computer. If the dbspace(s) and the transaction log are on the same disk, and a disk failure occurs, everything is lost. However, if the database and transaction log are stored on different disks, then most, if not all, the data can be recovered in the event of a disk failure because you have the full database or the transaction log (from which the database can be recovered).

The timestamp of a database or transaction log file is updated only when the file grows or when it is closed. If database operations cause the transaction log file to grow without the database file growing, the timestamp of the transaction log file is more recent than the timestamp of the database file. If the database is shut down, the transaction log file and the database timestamps are updated.

> ⚠ Caution
>
> The database file and the transaction log file must be located on the same physical computer as the database server or accessed via a SAN or iSCSI configuration. Database files and transaction log files located on a remote network directory can lead to poor performance, data corruption, and server instability.

When Changes Are Forced to Disk

Like the database file, the transaction log is organized into **pages**: fixed size areas of memory. When a change is recorded in the transaction log, it is made to a page in memory. The change is forced to disk when the earlier of the following operations happens:

- The page is full.
- A COMMIT is executed.

Completed transactions are guaranteed to be stored on disk, while performance is improved by avoiding a write to the disk on every operation.

Use the cooperative_commits and delayed_commits options to tune the precise behavior of the transaction log.

In this section:

Related Information

Recover from Media Failure [page 983]
Running a SQL Anywhere Database File that is Stored Remotely from the Server Machine
-m Database Server Option [page 465]
-m Database Option [page 553]
sa_disk_free_space System Procedure
delete_old_logs Option [MobiLink][SQL Remote] [page 729]

cooperative_commits Option [Deprecated] [page 713]
delayed_commits Option [page 727]

1.2.3.4.1 Transaction Log Validation

When a database using a transaction log mirror starts up, the database server performs a transaction log validation.

To do this, a series of checks and automatic recovery operations are performed to confirm that the transaction log and its mirror are not corrupt, and to correct some problems if corruption is detected.

On startup, the server checks that the transaction log and its mirror are identical by performing a full comparison of the two files; if they are identical, the database starts as usual. The comparison of log and mirror adds to database startup time.

If the database stopped because of a system failure, some operations might be written into the transaction log but not into the mirror. If the server finds that the transaction log and the mirror are identical up to the end of the shorter of the two files, the remainder of the longer file is copied into the shorter file. This produces an identical log and mirror. After this automatic recovery step, the server starts as usual.

If the check finds that the transaction log and the transaction log mirror are different in the body, one of the two files is corrupt. In this case, the database does not start, and an error message is generated saying that the transaction log or its mirror is invalid.

You can also use the Log Translation utility (dbtran) to validate transaction logs whether you have an online or offline transaction log. If the Log Translation utility can successfully read the transaction log file, it is valid.

Related Information

Log Translation Utility (dbtran) [page 1187]

1.2.3.4.2 Changing the Location of a Transaction Log (SQL Central)

Change the location of a transaction log in SQL Central.

Prerequisites

The database cannot be running when you change the location of the transaction log.

> ⚠ Caution
>
> The database file and the transaction log file must be located on the same physical computer as the database server or accessed via a SAN or iSCSI configuration. Database files and transaction log files

located on a remote network directory can lead to poor performance, data corruption, and database server instability.

For best results, the transaction log should be kept on a different disk from the database files.

Context

Perform this task when you need to move the transaction log to a different drive on the same computer.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. Click *Tools* > *SQL Anywhere 17* > *Change Log File Settings*.
3. Follow the instructions in the *Change Log File Settings Wizard*.

Results

The location of the transaction log is changed.

Related Information

Running a SQL Anywhere Database File that is Stored Remotely from the Server Machine
Transaction Log Utility (dblog) [page 1231]

1.2.3.4.3 Changing the Location of a Transaction Log (Command Line)

Change the location of a transaction log using the dblog utility.

Prerequisites

The database cannot be running when you change the location of the transaction log.

Context

Perform this task when you need to move the transaction log to a different drive on the same computer.

Procedure

Run the following command:

```
dblog -t new-transaction-log-file database-file
```

Results

The location of the transaction log is changed.

Related Information

Transaction Log Utility (dblog) [page 1231]

1.2.3.4.4 Renaming or Truncating the Transaction Log (SQL Central)

Rename or truncate the transaction log in SQL Central.

Prerequisites

You must have the BACKUP DATABASE privilege.

Context

If your database is involved in synchronization or replication, rename the transaction log to create a new transaction log for the database. If your database is not involved in replication and you have limited disk space on your computer, truncate the transaction log.

If your database is involved in synchronization or replication, it is good practice to maintain copies of old transaction logs until you are certain they are no longer needed. Alternatively to renaming the transaction log, you can continue to use the existing transaction log when you back up your database.

If you truncate the transaction log during backup to delete the contents of the online transaction log, to recover your database from media failure on the database file you must use every backup copy made since the last full backup.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. Right-click the database and click *Create Backup Images*.
3. Follow the instructions in the *Create Backup Images Wizard*.

Option	Action
Rename the transaction log	In the *What do you want to do with the transaction log* list, click *Rename the transaction log*.
Truncate the transaction log	In the *What do you want to do with the transaction log* list, click *Truncate the transaction log*.

Results

A backup is made and the transaction log is renamed or truncated.

Related Information

Database Recovery [page 960]
Backup Utility (dbbackup) [page 1121]
BACKUP DATABASE Statement

1.2.3.4.5 Renaming or Truncating the Transaction Log (SQL)

Rename or truncate the transaction log in SQL.

Prerequisites

You must have the BACKUP DATABASE system privilege.

Context

If your database is involved in synchronization or replication, rename the transaction log to create a new transaction log for the database. If your database is not involved in replication and you have limited disk space on your computer, truncate the transaction log.

If your database is involved in synchronization or replication, it is good practice to maintain copies of old transaction logs until you are certain they are no longer needed. Alternatively to renaming the transaction log, you can continue to use the existing transaction log when you back up your database.

If you truncate the transaction log during backup to delete the contents of the online transaction log, to recover your database from media failure on the database file you must use every backup copy made since the last full backup.

Procedure

Use the BACKUP DATABASE statement with the following clauses:

Option	Action
Rename the transaction log	`BACKUP DATABASE` `DIRECTORY backup-directory` `[ TRANSACTION LOG ONLY ]` `TRANSACTION LOG RENAME;`
Truncate the transaction log	`BACKUP DATABASE` `DIRECTORY backup-directory` `[ TRANSACTION LOG ONLY ]` `TRANSACTION LOG TRUNCATE;`

Include the TRANSACTION LOG ONLY clause only if you are making an incremental backup.

The backup copies of the transaction log and database file are placed in `backup-directory`. If you provide a path, it is relative to the working directory of the database server, not your client application.

Results

A backup is made and the transaction log is renamed or truncated.

Related Information

Database Recovery [page 960]
Backup Utility (dbbackup) [page 1121]
BACKUP DATABASE Statement

1.2.3.4.6 Renaming or Truncating the Transaction Log (Command Line)

Rename or truncate the transaction log from the command line.

Prerequisites

You must have the BACKUP DATABASE system privilege.

Context

If your database is involved in synchronization or replication, rename the transaction log to create a new transaction log for the database. If your database is not involved in replication and you have limited disk space on your computer, truncate the transaction log.

If your database is involved in synchronization or replication, it is good practice to maintain copies of old transaction logs until you are certain they are no longer needed. Alternatively to renaming the transaction log, you can continue to use the existing transaction log when you back up your database.

If you truncate the transaction log during backup to delete the contents of the online transaction log, to recover your database from media failure on the database file you must use every backup copy made since the last full backup.

Procedure

Run the following command:

Option	Action
Rename the transaction log	`dbbackup -c "connection-string" -r [ -t ] backup-directory`
Truncate the transaction log	`dbbackup -c "connection-string" -x [ -t ] backup-directory`

Include the -t option only if you are making an incremental backup.

The backup copies of the transaction log and database file are placed in `backup-directory`. If you specify a path, it is relative to the directory from which you run the command.

Results

A backup is made and the transaction log is renamed or truncated.

Related Information

Database Recovery [page 960]
Backup Utility (dbbackup) [page 1121]
BACKUP DATABASE Statement

1.2.3.4.7 Transaction Log Size Considerations

The size of the transaction log can affect recovery times.

You can control transaction log file growth by ensuring that all your tables have compact primary keys. If you perform updates or deletes on tables that do not have a primary key or a unique index not allowing NULL, the entire contents of the affected rows are entered in the transaction log. If a primary key is defined, the database server needs to store only the primary key column values to uniquely identify a row. If the table contains many columns or wide columns, the transaction log pages fill up much faster if no primary key is defined. In addition to taking up disk space, this extra writing of data affects performance.

If a primary key does not exist, the server looks for a UNIQUE NOT NULL index on the table (or a UNIQUE constraint). A UNIQUE index that allows NULL is not enough.

Related Information

-m Database Server Option [page 465]
-m Database Option [page 553]
sa_disk_free_space System Procedure

delete_old_logs Option [MobiLink][SQL Remote] [page 729]

1.2.3.4.8 Transaction Log Mirrors

A **transaction log mirror** is an identical copy of the transaction log, maintained at the same time as the transaction log.

If a database has a transaction log mirror, every database change is written to both the transaction log and the transaction log mirror. By default, databases do not have transaction log mirrors.

A transaction log mirror provides extra protection for critical data. It enables complete data recovery if a media failure on the transaction log occurs. A transaction log mirror also enables a database server to perform automatic validation of the transaction log on database startup.

It is recommended that you use a transaction log mirror when running high-volume or critical applications. For example, at a consolidated database in a SQL Remote setup, replication relies on the transaction log, and if the transaction log is damaged or becomes corrupt, data replication can fail.

If you are using a transaction log mirror, and an error occurs while trying to write to one of the logs (for example, if the disk is full), the database server stops. The purpose of a transaction log mirror is to ensure complete recoverability if a media failure occurs on either log device; this purpose would be lost if the server continued with a single transaction log.

You can specify the -fc option when starting the database server to implement a callback function when the database server encounters a file system full condition.

Where to Store the Transaction Log Mirror

There is a performance penalty for using a transaction log mirror because each database log write operation must be performed twice. The performance penalty depends on the nature and volume of database traffic and on the physical configuration of the database and logs.

A transaction log mirror should be kept on a separate device from the transaction log. This improves performance, and if either device fails, the other copy of the log keeps the data safe for recovery.

Alternatives to a Transaction Log Mirror

Alternatives to a transaction log mirror are to use the following configurations:

- database mirroring.
- a disk controller that provides hardware mirroring. Generally, hardware mirroring is more expensive than operating-system level software mirroring, but it provides better performance.
- operating-system level software mirroring, as provided by Microsoft Windows.

Live backups provide additional protection with some similarities to using a transaction log mirror.

In this section:

Related Information

Database Mirroring [page 1757]
Live Backups (Continuous Backups) [page 944]
-fc Database Server Option [page 425]
Initialization Utility (dbinit) [page 1166]
Transaction Log Utility (dblog) [page 1231]

1.2.3.4.8.1 Starting a Transaction Log Mirror for an Existing Database (SQL Central)

Start a transaction log mirror for an existing database in SQL Central.

Prerequisites

You must be able to access the directories where the transaction log is located.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. Click *Tools* > *SQL Anywhere 17* > *Change Log File Settings*.
3. Follow the instructions in the *Change Log File Settings Wizard*.

Results

A transaction log mirror is started for the database.

Related Information

Transaction Log Utility (dblog) [page 1231]

1.2.3.4.8.2 Starting a Transaction Log Mirror for an Existing Database (Command Line)

Start a transaction log mirror for an existing database (command line).

Prerequisites

You must be able to access the directories where the transaction log is located.

Context

Maintain the transaction log mirror for an existing database any time the database is not running. You can also use the dblog utility to stop a database from using a transaction log mirror.

Procedure

1. Ensure that the database is not running.
2. Run the following command:

```
dblog -m mirror-file database-file
```

Results

A transaction log mirror is started for the database.

Related Information

Transaction Log Utility (dblog) [page 1231]

1.2.3.4.9 Determining Which Connection Has an Outstanding Transaction (SQL)

Determine which user has outstanding transactions in SQL.

Prerequisites

You must have the SERVER OPERATOR, MONITOR, or DROP CONNECTION system privilege.

Context

If you are performing a backup that renames or deletes the transaction log, incomplete transactions are carried forward to the new transaction log.

Procedure

1. Connect to the database from Interactive SQL.
2. Run the sa_conn_info system procedure:

```
CALL sa_conn_info;
```

Results

Inspect the *UncommitOps* column in the *Results* pane to see which connection has uncommitted operations.

Next Steps

If necessary, you can disconnect the user with a DROP CONNECTION statement.

Related Information

sa_conn_info System Procedure
SQL Anywhere Cockpit [page 1419]
DROP CONNECTION Statement

1.2.3.4.10 Checkpoints

A **checkpoint** writes dirty pages to disk and represents a known consistent state of the database on disk.

Following a checkpoint, the contents of the checkpoint log are deleted. The empty checkpoint log pages remain in the checkpoint log within the current session and can be reused for new checkpoint log data. As the checkpoint log increases in size, so does the database file.

At a checkpoint, all the data in the database is held on disk in the database file. The information in the database file matches that in the transaction log. During recovery, the database is first recovered to the most recent checkpoint, and then changes since that checkpoint are applied.

At the end of each session, a history of the checkpoint log usage is maintained in the database and is used to determine an appropriate size for the checkpoint log for the next session.

The database server can initiate a checkpoint and perform other operations while the checkpoint takes place. However, if a checkpoint is already in progress, then any statement that triggers a new checkpoint must wait for the current checkpoint to finish before it can be executed.

Statements That Trigger Implicit or Explicit Checkpoints

- ALTER INDEX REBUILD statement
- ALTER Table statement (unless the ALTER TABLE statement adds a nullable column or a referential integrity constraint and does not physically modify the table's pages)
- BACKUP DATABASE statement
- CHECKPOINT statement
- COMMIT statement (when the database does not have a transaction log)
- CREATE DBSPACE statement
- CREATE INDEX statement (when used to create an index on a function (an implicit computed column))
- DROP DBSPACE statement
- DROP TABLE statement and DROP MATERIALIZED VIEW statement (only if the table or view contains at least one row)
- LOAD TABLE statement
- REFRESH MATERIALIZED VIEW statement
- REORGANIZE TABLE statement

In this section:

How the Database Server Decides When to Checkpoint [page 306]
There are several conditions under which the database server performs a checkpoint.

Checkpoint Logs [page 307]
The **checkpoint log** is located at the end of the database file and is stored in the system dbspace.

Related Information

Database Backup [page 940]
ALTER INDEX Statement
ALTER TABLE Statement
BACKUP DATABASE Statement
CHECKPOINT Statement
COMMIT Statement
CREATE DBSPACE Statement
CREATE INDEX Statement
DROP DBSPACE Statement
DROP TABLE Statement
DROP MATERIALIZED VIEW Statement
LOAD TABLE Statement
REFRESH MATERIALIZED VIEW Statement
REORGANIZE TABLE Statement

1.2.3.4.10.1 How the Database Server Decides When to Checkpoint

There are several conditions under which the database server performs a checkpoint.

When a database shuts down cleanly, the database file holds a complete and current copy of all the data in the database. When a database is running, however, the database file is generally not current or complete. The only time a database file is guaranteed to hold a complete and current copy of all data is immediately after a checkpoint completes. Following a checkpoint, all the contents of the database cache are on disk.

The database server checkpoints a database under the following conditions:

- As part of the database shutdown operations
- When the amount of time since the last checkpoint exceeds the setting of the -gc server option
- When the estimated time to do a recovery operation exceeds the setting of the -gr server option
- When the database server is idle long enough to write all dirty pages
- When certain DDL statements (such as ALTER TABLE, DROP TABLE, DROP INDEX, LOAD TABLE, or BACKUP) are executed
- When a connection issues a CHECKPOINT statement
- When the database server is running without a transaction log and a transaction is committed

The priority of writing dirty pages to the disk increases as the time and the amount of work since the last checkpoint increases. The priority is determined by the following factors:

Checkpoint Urgency

The time that has elapsed since the last checkpoint, as a percentage of the checkpoint time setting of the database. You can set the maximum time, in minutes, between checkpoints by using the -gc server option or the checkpoint_time database option. If -gc is specified, the checkpoint_time option setting in the database is ignored.

Recovery Urgency

A heuristic to estimate the amount of time required to recover the database if it fails right now. You can set the maximum time, in minutes, for recovery in the event of system failure by using the -gr server option or recovery_time database option. If -gr is specified, the recovery_time option setting in the database is ignored.

The checkpoint and recovery urgency values are important only if the database server does not have enough idle time to write dirty pages. The lower boundary on the interval between checkpoints is based on a combination of the recovery_time and checkpoint_time options. The recovery_time option setting is not respected when it would force a checkpoint too soon.

Frequent checkpoints make recovery quicker, but also create work for the server writing out dirty pages.

If, because of other activity in the database, the number of dirty pages falls to zero, and if the checkpoint urgency is 33% or more, then a checkpoint takes place automatically since it is a convenient time.

Both the checkpoint urgency and recovery urgency values increase until the checkpoint occurs, at which point they drop to zero.

Related Information

Checkpoint Logs [page 307]
-gc Database Server Option [page 429]
checkpoint_time Option [page 700]
-gr Database Server Option [page 444]
recovery_time Option [page 806]

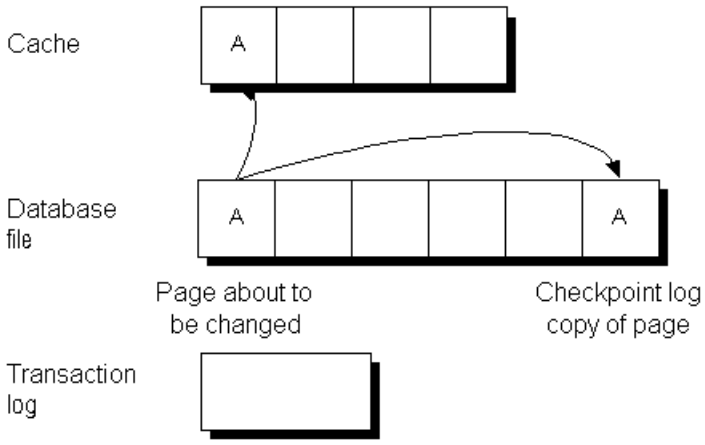
1.2.3.4.10.2 Checkpoint Logs

The **checkpoint log** is located at the end of the database file and is stored in the system dbspace.

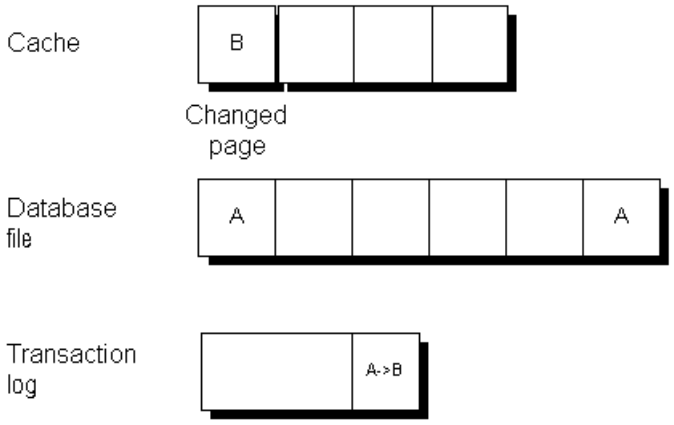
The database file is composed of pages: fixed size portions of hard disk. Pages are added to the checkpoint log as necessary during a session, and at the end of the session, a history of the checkpoint log usage is stored in the database. This history is used to determine an appropriate size for the checkpoint log in future sessions.

Before any page is updated (made **dirty**), the database server performs the following operations:

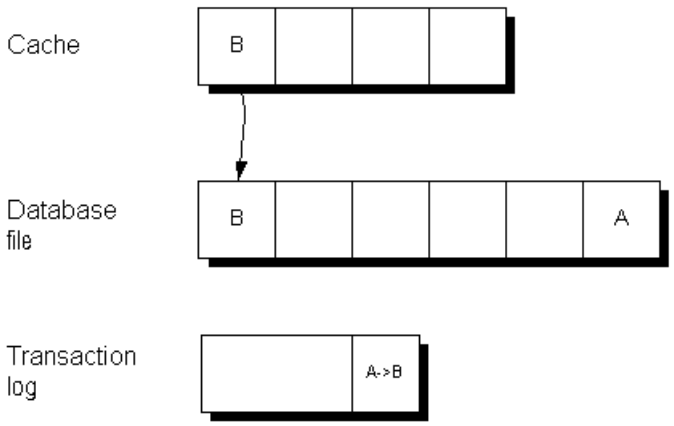
- It reads the page into memory, where it is held in the database cache.
- It makes a copy of the original page. These copied pages are the checkpoint log.



Changes made to the page are applied to the copy in the cache. For performance reasons they are not written immediately to the database file on disk.



When the cache is full, the changed page may get written out to disk. The copy in the checkpoint log remains unchanged.



Related Information

Database Backup [page 940]
How the Database Server Decides When to Checkpoint [page 306]

1.2.3.4.11 Offline Transaction Log

In addition to backing up the transaction log, a backup operation can rename the online transaction log to a file name of the form `YYMMDDxx.log`. This is called an **offline** transaction log.

This file is no longer used by the database server, but is available for the Message Agent. A new online transaction log is started with the same name as the old online transaction log.

The `YYMMDDxx.log` file names are used to distinguish between the files, not for ordering. For example, the renamed log file from the first backup on December 10, 2000, is named `001210AA.log`. The first two digits indicate the year, the second two digits indicate the month, the third two digits indicate the day of the month, and the final two characters distinguish among different backups made on the same day.

The Message Agent can use the offline copies to provide the old transactions as needed. If you set the delete_old_logs database option to On, then the Message Agent deletes the offline files when they are no longer needed, saving disk space.

1.2.3.5 The Utility Database (utility_db)

The **utility database** is a phantom database with no physical representation.

The utility database has no database file, and therefore it cannot contain data. This feature allows you to:

- Execute database file administration statements such as CREATE DATABASE without first connecting to an existing physical database.
- Retrieve values of connection properties and server properties using the utility database.
- Connect to a running database server when it is not possible to connect to the database. For example, in a mirroring system, you can connect to the utility database to stop a mirroring server or force a database to become the primary server.

The utility database is named utility_db. If you attempt to create a database with this name, the operation fails.

Use the -su database server option to set the utility database password, and optionally the user ID, or to disable connections to the utility database. The default user ID for the utility database is DBA. Passwords for the utility database must be at least six characters long and are case sensitive.

You can start the utility database on a database server by specifying utility_db as the database name when connecting to the server.

When connected to the utility database, only the following statements can be executed:

- ALTER DATABASE `dbfile` ALTER TRANSACTION LOG.
- ALTER DATABASE `database-name` FORCE START

- CREATE DATABASE statement.
- CREATE DECRYPTED DATABASE statement.
- CREATE DECRYPTED FILE statement.
- CREATE ENCRYPTED DATABASE statement.
- CREATE ENCRYPTED FILE statement.
- DROP DATABASE statement.
- CREATE USER DBA IDENTIFIED BY `new-password`.
- RESTORE DATABASE statement.
- REVOKE CONNECT FROM DBA.
- SET TEMPORARY OPTION progress_messages = [OFF | RAW | FORMATTED].
- SELECT statement without a FROM or WHERE clause.
- START DATABASE statement.
- STOP DATABASE statement.
- STOP SERVER statement.

The -gu database server option sets the privilege required for executing database file administration statements. When -gu utility_db is specified, only the utility database can execute the statements.

Example

Executing the following statement after connecting to the utility database creates a database named `new.db` in the directory `c:\temp`.

```
CREATE DATABASE 'c:\\temp\\new.db' DBA USER 'DBA' DBA PASSWORD 'passwd';
```

Executing the following statement on the utility database returns the default collation sequence that is used when creating a database:

```
SELECT PROPERTY( 'DefaultCollation' );
```

In this section:

Connecting to the Utility Database (Connect Window) [page 311]
Connect to the utility database to execute database file administration statements, to query connection and server properties, and to connect to a running database server when it is not possible to connect to the database.

Related Information

Stopping a Database Server in a Mirroring System (dbstop Utility) [page 1781]
Troubleshooting: The Primary Server Cannot be Restarted [page 1792]
Hiding the Contents of an .ini File [page 591]
-gu Database Server Option [page 452]

-su Database Server Option [page 502]

1.2.3.5.1 Connecting to the Utility Database (Connect Window)

Connect to the utility database to execute database file administration statements, to query connection and server properties, and to connect to a running database server when it is not possible to connect to the database.

Prerequisites

The -su database server option specifies the password, and, optionally, the user ID for the utility database. To connect to a utility database on a server that was started using the -su option, you need the user ID and password that was set by the -su option.

If the -su database server option was not specified for a network server, then you cannot connect to the utility database.

If the -su database server option was not specified for a personal database server, you can connect to the utility database and you do not need a password.

Context

You have a running database server and you want to connect to its utility database.

If the database database server is a network server (dbsrv17), then it was started using the -su option and you have this password and (optionally) user ID.

```
dbsrv17 -n server-name -su user-ID,password
```

If the database server is a personal database server (dbeng17), then either it was started using the -su option and the utility database has a user ID (optional) and password, or it was started without the -su option and the utility database has no user ID and password.

```
dbeng17 -n server-name
```

Procedure

1. Open the *Connect* window
2. In the *User ID* field, choose one of the following options:

Option	Action
If the -su database server option was specified and included a user ID	Type the password specified by the -su database server option
If the -su database server option was not specified, or -su specified only a password	Type DBA

3. In the *Password* field choose one of the following options:

Option	Action
Personal server (dbeng17) and no -su database server option	Type any password that is at least 6 characters in length.
Network server (dbsrv17)	Type the password specified by the -su option.

4. In the *Action* dropdown list, click *Connect To A Running Database On This Computer*.
5. In the *Server Name* field, type `server-name`.
6. In the *Database Name* field, type **`utility_db`**.
7. Click *Connect*.

Results

Interactive SQL connects to the utility database on the personal or network server.

When you are connected to the utility database, executing REVOKE CONNECT FROM `user-ID` disables future connections to the utility database. No future connections can be made to the utility database unless you use a connection that existed before the REVOKE CONNECT was done, or restart the database server.

Related Information

Database Connections [page 5]
Stopping a Database Server in a Mirroring System (dbstop Utility) [page 1781]
Troubleshooting: The Primary Server Cannot be Restarted [page 1792]
-su Database Server Option [page 502]
File Hiding Utility (dbfhide) [page 1160]
REVOKE Statement

1.2.3.6 Dropping a Database (SQL Central)

Drop a database in SQL Central.

Prerequisites

The database to be dropped must not be running when the *Erase Database Wizard* is used.

Context

Dropping a database deletes all tables and data from disk, including the transaction log that records alterations to the database.

All database files are read-only to prevent accidental modification or deletion of database files.

Procedure

1. In SQL Central, click *Tools* > *SQL Anywhere 17* > *Erase Database*.
2. Follow the instructions in the wizard.

Results

The database is dropped.

Related Information

Connecting to the Utility Database (Connect Window) [page 311]
-gu Database Server Option [page 452]
Erase Utility (dberase) [page 1155]
DROP DATABASE Statement

1.2.3.7 Dropping a Database (dberase Utility)

Drop a database by using the dberase utility.

Prerequisites

The database to be dropped must not be running when the dberase utility is used.

Context

Dropping a database deletes all tables and data from disk, including the transaction log that records alterations to the database.

All database files are read-only to prevent accidental modification or deletion of database files.

The dberase utility does not drop dbspaces. To drop a dbspace, use the DROP DBSPACE statement.

Procedure

Run the dberase utility.

Results

The database has been erased.

Example

The following command drops the temp database.

```
dberase c:\temp\temp.db
```

Related Information

Connecting to the Utility Database (Connect Window) [page 311]
-gu Database Server Option [page 452]

Erase Utility (dberase) [page 1155]
DROP DBSPACE Statement

1.2.4 Column Data Type Considerations

There are several data types available for columns.

- Binary data types
- Character data types
- Date/time data types
- Decimal data types
- Domains (user-defined data types)
- Floating-point data types
- Integer data types
- Spatial data types

Any of the character or binary string data types such as CHAR, VARCHAR, LONG VARCHAR, NCHAR, BINARY, VARBINARY, and so on, can be used to store large objects such as images, word-processing documents, and sound files.

In this section:

NULL and NOT NULL Columns [page 315]
If the column value is mandatory for a row, you define the column as being NOT NULL; otherwise, the column is allowed to contain the NULL value, which represents no value.

BLOB Considerations [page 316]
A BLOB is an uninterpreted string of bytes or characters, stored as a value in a column.

Related Information

SQL Data Types

1.2.4.1 NULL and NOT NULL Columns

If the column value is mandatory for a row, you define the column as being NOT NULL; otherwise, the column is allowed to contain the NULL value, which represents no value.

The default is to allow NULL values, but you should explicitly declare columns NOT NULL unless there is a good reason to allow NULL values.

The sample database has a table called Departments, which has columns named DepartmentID, DepartmentName, and DepartmentHeadID. Its definition is as follows:

Column	Data type	Size	NULL/NOT NULL	Constraint
DepartmentID	integer	-	NOT NULL	Primary key
DepartmentName	char	40	NOT NULL	None
DepartmentHeadID	integer	-	NULL	Foreign key

If you specify NOT NULL, a column value must be supplied for every row in the table.

Related Information

NULL Special Value
Search Conditions

1.2.4.2 BLOB Considerations

A BLOB is an uninterpreted string of bytes or characters, stored as a value in a column.

Common examples of a BLOB are picture or sound files. While BLOBs are typically large, you can store them in any character string or binary string data type such as CHAR, VARCHAR, NCHAR, BINARY, VARBINARY, and so on. Choose your data type and length depending on the content and length of BLOBs you expect to store.

> **i Note**
>
> While a character large object is commonly called a CLOB, a binary large object is called a BLOB, and the combination of both is called a LOB. Only the acronym BLOB is used in this documentation.

When you create a column for storing BLOB values using the CREATE TABLE statement, you can control aspects of their storage. For example, you can specify that BLOBs up to a specified size be stored in the row (inline), while larger BLOBs are stored outside the row in table extension pages. Additionally, you can specify that for BLOBs stored outside the row, the first `n` bytes of the BLOB, also referred to as the prefix, are duplicated in the row. These storage aspects are controlled by the INLINE and PREFIX settings specified in the CREATE TABLE and ALTER TABLE statements. The values you specify for these settings can have unanticipated impacts on performance or disk storage requirements.

If neither INLINE nor PREFIX is specified, or if INLINE USE DEFAULT or PREFIX USE DEFAULT is specified, default values are applied as follows:

- For character data type columns, such as CHAR, NCHAR, LONG VARCHAR, and XML, the default value of INLINE is 256, and the default value of PREFIX is 8.
- For binary data type columns, such as BINARY, LONG BINARY, VARBINARY, BIT, VARBIT, LONG VARBIT, BIT VARYING, and UUID, the default value of INLINE is 256, and the default value of PREFIX is 0.

It is recommended that you do not set INLINE and PREFIX values unless there are specific requirements for which the default values are insufficient. The default values have been chosen to balance performance and disk space requirements. For example, row processing performance may degrade if you set INLINE to a large value,

and all the BLOBs are stored inline. If you set PREFIX too high, you increase the amount of disk space required to store BLOBs since the prefix data duplicates a portion of the BLOB.

If you do decide to set INLINE or PREFIX values, the INLINE length must not exceed the length of the column. Likewise, the PREFIX length, must not exceed the INLINE length.

The prefix data for a compressed column is stored uncompressed, so if all the data required to satisfy a request is stored in the prefix, no decompression is necessary.

BLOB Sharing

If a BLOB exceeds the inline size, and requires more than one database page for storage, the database server stores it so that it can be referenced by other rows in the same table, when possible. This is known as BLOB sharing. BLOB sharing is handled internally and is intended to reduce unnecessary duplication of BLOBs in the database.

BLOB sharing only occurs when you set values of one column to be equal to those of another column. For example, `UPDATE t column1=column2;`. In this example, if column2 contains BLOBs, instead of duplicating them in column1, pointers to the values in column2 are used instead.

When a BLOB is shared, the database server keeps track of how many other references there are to the BLOB. Once the database server determines that a BLOB is no longer referenced within a table, the BLOB is removed.

If a BLOB is shared between two uncompressed columns and one of those columns is then compressed, the BLOB will no longer be shared.

Related Information

CREATE TABLE Statement

1.2.5 Column Compression Considerations

CHAR, VARCHAR, and BINARY columns can be compressed to save disk space.

For example, you can compress a column in which large BLOB files such as BMPs and TIFFs are stored. Compression is achieved using the deflate compression algorithm. This is the same algorithm used by the COMPRESS function, and is also the same algorithm used for Windows ZIP files.

Compressed columns can reside inside of encrypted tables. In this case, data is first compressed, and then encrypted.

If a string compresses to a value that is not at least one page smaller than the original value, it is stored uncompressed. Also, a string that is not longer than the column INLINE value is stored uncompressed.

To compress columns, use the COMPRESSED clause of the CREATE TABLE and ALTER TABLE statements.

You can determine the benefits you are getting by compressing columns using the sa_column_stats system procedure.

Related Information

Table Encryption [page 1706]
CREATE TABLE Statement
ALTER TABLE Statement
sa_column_stats System Procedure

1.2.6 Constraint Considerations

Although the data type of a column restricts the values that are allowed in that column (for example, only numbers or dates), you may want to further restrict the allowed values.

You can restrict the values of any column by specifying a CHECK constraint. A CHECK constraint is a restriction that enforces specified conditions on a column or set of columns. You can use any valid condition that could appear in a WHERE clause to restrict the allowed values. Most CHECK constraints use either the BETWEEN or IN condition.

Related Information

Data Integrity
CHECK Constraints on Columns
CHECK Constraints on Tables
Search Conditions

1.3 Database Servers

Two versions of the SQL Anywhere database server are provided: the **personal server** (dbeng17) and the **network server** (dbsrv17).

SQL Anywhere Database Servers

A database created with a personal database server can be used with a network database server and vice versa. The request-processing engine is identical in both the personal and network servers and each one supports exactly the same SQL language and many of the same database features. However, there are a few differences between the two servers.

The Personal Database Server

This executable is provided for single-user, same-computer use (for example, as an embedded database server) and does not support client/server communications across a network.

On Windows operating systems, the name of the personal server executable is *dbeng17*`.exe`. On Unix operating systems its name is *dbeng17*.

The Network Database Server

This executable supports client/server communications across a network, and is intended for multi-user use.

On Windows operating systems, the name of the network server executable is *dbsrv17*`.exe`. On Linux and Unix operating systems, the name is *dbsrv17*.

Network Software Requirements

If you are running a network server, you must have appropriate networking software installed and running.

The TCP/IP network protocol is supported.

In this section:

How to Suppress Microsoft Windows Event Log Messages [page 340]
You can suppress Windows event log entries by setting a registry entry.

Special Modes [page 341]
You can run SQL Anywhere in special modes.

How to Control Performance and Memory [page 342]
Several settings can affect database server performance.

Threading [page 344]
To understand the threading model, you must understand the basic terminology and concepts of threading and request processing.

How to Run the Database Server as a Service or Daemon [page 354]
There are several ways in which the database server can be run as a service or daemon.

Authenticated Applications for OEM Editions [page 369]
The OEM Edition of SQL Anywhere can perform any operation on the database.

SQL Anywhere on Windows [page 379]
There are several considerations that apply when running SQL Anywhere software on Windows.

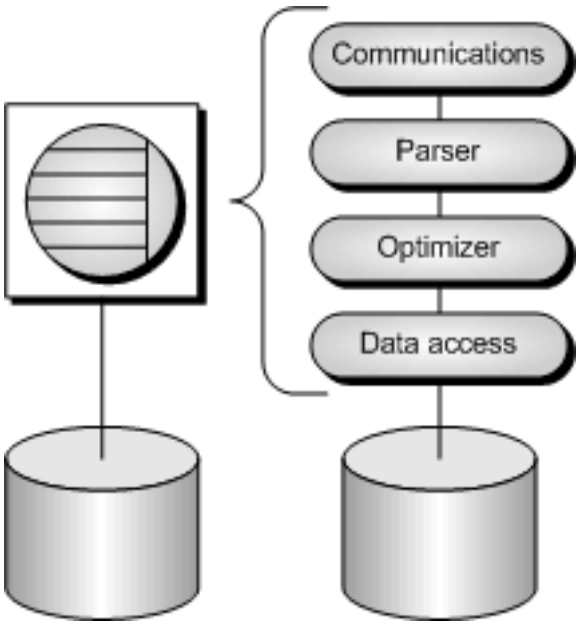
Related Information

SQL Anywhere Database Server Executable (dbsrv17, dbeng17) [page 380]

1.3.1 Inside the SQL Anywhere Database Server

The SQL Anywhere database server has an internal structure that allows many requests to be handled efficiently.

- A communications layer handles the exchange of data with client applications. This layer receives requests from client applications, and returns results. The timing of these actions is governed by a negotiation between the client and the server to make sure that the network traffic is kept to a minimum, but that the data is made available as soon as possible on the client side.
- The parser checks each SQL statement sent to the database server, and transforms it into an internal form for processing.
- If the request is a query, an update, or delete statement, there can be many different ways of accessing the data, which may take significantly different times. The optimizer selects the best method of getting the required data quickly.
- The lowest level of the database server is concerned with reading and writing data from the disk, caching data in memory to avoid unnecessary disk access, and balancing the demands of different users.



1.3.2 Differences Between the Network (dbsrv17) and Personal (dbeng17) Servers

There are common database server features that differ between personal database servers and network database servers.

The personal database server has a maximum of ten concurrent connections, uses at most four cores on one CPU for request processing, and doesn't support network client/server connections.

By default, the personal database server only uses the shared memory protocol. You must use the -x option to use TCP/IP with the personal database server.

In addition, there are other minor differences, such as the default privilege level that is required to start new databases, or the privileges required to execute the CHECKPOINT statement.

Feature	Network database server (dbsrv17)	Personal database server (dbeng17)
Checkpoints	The CHECKPOINT system privilege is required to use the CHECKPOINT statement.	No system privileges are required to use the CHECKPOINT statement.
Communication packet compression	Supported.	Unsupported.
Connecting to the utility database	By default, no connections are allowed to the utility database. You must use the -su option to specify the password for connecting to the utility database.	By default, connections to the utility database are allowed with the user ID DBA and any password.
Connection types	Shared memory and TCP/IP by default.	Shared memory by default. To connect with the TCP/IP protocol use the -x option.

Feature	Network database server (dbsrv17)	Personal database server (dbeng17)
Database mirroring	Supported.	Unsupported.
LDAP as a name server	Can only be used with TCP/IP.	Unsupported. Do not confuse this with LDAP user authentication which is supported.
Multiprogramming level tuning	Supported.	Unsupported.
Network communications	Supported.	Unsupported.
Number of connections	Limited by license.	Maximum of ten simultaneous connections, of which three are reserved for standard connections..
Number of CPUs	Limited by license.	Maximum of four cores on one CPU.
Read-only scale-out	Supported.	Unsupported.

Database Server and Database Startup Options

The following database server and database startup options are supported *only* by the network database server.

- -gna database server option
- -gnh database server option
- -gnl database server option
- -gns database server option
- -pc database server option
- -pt database server option
- -sn database option
- -xa database server option
- -xf database server option
- -xp database option

Database server properties

The following database server properties are supported *only* by the network database server.

- AutoMultiProgrammingLevel server property
- AutoMultiProgrammingLevelStatistics server property
- MaxMultiProgrammingLevel server property
- MinMultiProgrammingLevel server property
- UniqueClientAddresses server property

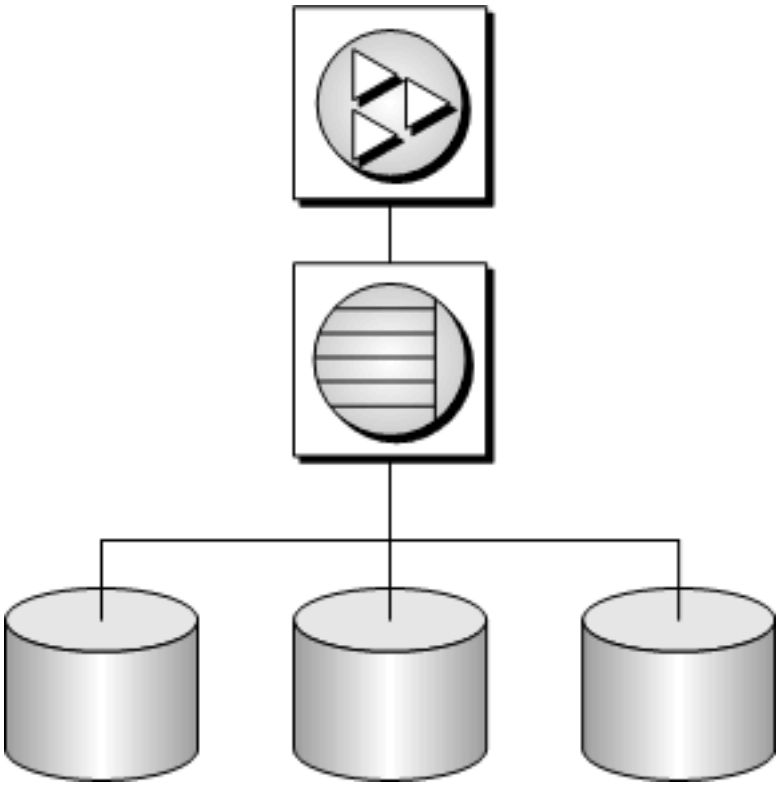
Related Information

Database Server Startup Options [page 390]
Alphabetical List of Database Options [page 663]
Database Mirroring [page 1757]
Connections Using LDAP as a Name Server [page 181]
Database Server Configuration of the Multiprogramming Level [page 348]
Editions and Licensing
Read-only Scale-out [page 1830]

1.3.3 Multiple Databases Running on a Single Database Server

Both the personal database server and the network database server can manage many databases simultaneously.

Each connection from an application must be to a single database, but an application can use separate connections to different databases, or a set of applications can work on different databases, all through the same database server.



Databases can be started when the database server is started, by connecting to a database using the DatabaseFile (DBF) connection parameter, or by using the START DATABASE statement.

In this section:

1.3.3.1 Data Access to Other Databases

You can access databases on multiple database servers, or even on the same server, using remote data access.

The application is still connected to a single database, but by defining remote servers, you can use proxy tables that exist on the remote database as if they were in the database to which you are connected.

1.3.4 How to Start the Database Server

Starting a database server varies depending on your operating system.

As well, you can specify commands in several ways, depending on your operating system.

- Run the command at a command prompt.
- Place the command in a shortcut or desktop icon.
- Run the command in a batch file.
- Include the command as a StartLine (START) connection parameter in a connection string.

There are slight variations in how you specify the basic command from platform to platform.

> **i** Note
>
> - Except where otherwise noted, these commands start the network server (***dbsrv17***). To start a personal server, replace ***dbsrv17*** with ***dbeng17***.
> - If the database file is in the starting directory for the command, you do not need to specify `path`.
> - If you do not specify a file extension in `database-file`, the extension `.db` is assumed.

Example

Command	Comments
On Windows, click *Start* > *Programs* > *SQL Anywhere 17* > *SQL Anywhere* > *Personal Server*	The *Server Startup Options* window appears where you can specify information to start a personal database server (dbeng17 executable). You can also start a database on the database server.

Command	Comments
On Windows, click *Start* > *Programs* > *SQL Anywhere 17* > *SQL Anywhere* > *Network Server*	The *Server Startup Options* window appears where you can specify information to start a network database server (dbsrv17 executable). You can also start a database on the database server.
`dbsrv17 demo`	Use a database file name in a connection string. Run this command in the directory where *demo.db* is located to start both a network server and a database called *demo.db*:
`dbsrv17 path\database-file`	This command starts the database server on Windows. If you omit the database file, the *Server Startup Options* window appears where you can locate a database file by clicking *Browse*.

Related Information

Network Protocol Options [page 187]
SQL Script Files
-x Database Server Option [page 523]
StartLine (START) Connection Parameter [page 112]
SQL Anywhere Database Server Executable (dbsrv17, dbeng17) [page 380]
Troubleshooting: Database Server Startup [page 1023]

1.3.5 How to Stop the Database Server

There are several ways to stop the database server.

- Clicking *Shut Down* on the database server messages window.
- Using the dbstop utility.
  The dbstop utility is useful in batch files, or for stopping a server on another computer. It requires a connection string in its command.
- Letting it shut down automatically by default when the application disconnects.
- Pressing Q when the database server messages window has the focus on Unix.

In this section:

Who Can Stop a Database Server? [page 326]
When you start a database server, use the -gk option to set the level of privileges required for users to stop the server with dbstop.

Operating System Session Shutdown [page 326]

If you close an operating system session where a database server is running, or if you use an operating system command to stop the database server, the server shuts down, but not cleanly.

Stopping a Database Server (dbstop Utility) [page 327]
Stop a database server using the dbstop utility.

1.3.5.1 Who Can Stop a Database Server?

When you start a database server, use the -gk option to set the level of privileges required for users to stop the server with dbstop.

For personal database servers, the default is all. The default level of system privilege required is SERVER OPERATOR for network database servers, but you can also set the value to all or none. (However, anyone at the computer can click *Shut Down* on the database server messages window.)

Related Information

-gk Database Server Option [page 433]

1.3.5.2 Operating System Session Shutdown

If you close an operating system session where a database server is running, or if you use an operating system command to stop the database server, the server shuts down, but not cleanly.

The next time the database loads, recovery is required, and happens automatically.

It is better to stop the database server explicitly before closing the operating system session.

Examples of commands that do not stop a server cleanly include:

- Stopping the process in the Windows Task Manager
- Using a UNIX or Linux slay or kill command

Related Information

Database Backup and Recovery [page 939]

1.3.5.3 Stopping a Database Server (dbstop Utility)

Stop a database server using the dbstop utility.

Prerequisites

By default, to stop a running network server, you must have the SERVER OPERATOR system privilege. The required privileges can be changed by using the -gk database server option.

By default, any user can stop a personal database servers.

However, anyone at the computer can click *Shut Down* on the *Database Server Messages* window.

Context

The dbstop utility is useful in batch files, or for stopping a server on another computer. It requires a connection string in its command.

Procedure

At a command prompt, execute a statement similar to the following using the dbstop utility, where `server-name` is the name of the server you want to stop, and `user-ID` and `password` are the connection parameters:

```
dbstop -c "Server=server-name;UID=user-ID;PWD=password"
```

Results

The database server is stopped.

Example

Start a database server. Run the following command from the SQL Anywhere installation directory to start a server named Ottawa using the sample database:

```
dbsrv17 -n Ottawa "%SQLANYSAMP17%\demo.db"
```

Stop the server using dbstop:

```
dbstop -c "Server=Ottawa;UID=DBA;PWD=sql"
```

Related Information

-gk Database Server Option [page 433]
Stop Server Utility (dbstop) [page 1219]

1.3.6 Database Starting and Stopping

A database server can have more than one database loaded at a time.

You can start databases and start the database server at the same time, as follows:

```
dbsrv17 demo sample
```

> ⚠ Caution
>
> The database file must be on the same computer as the database server. Managing a database file that is located on a network drive can lead to file corruption.

Starting a Database on a Running Server

You can also start databases after starting a server in one of the following ways:

- Connect to a database using a DatabaseFile (DBF) connection parameter while connected to a server. The DatabaseFile (DBF) connection parameter specifies a database file for a new connection. The database file is started on the current server.
- Use the START DATABASE statement.
- In SQL Central, select a server, and then click *File* > *Start Database*.

Limitations

- The server holds database information in memory using pages of a fixed size. Once a server has been started, you cannot start a database that has a larger page size than the server.
- The -gd server option decides the privileges required to start and stop databases.

In this section:

Starting a Database Without Connecting (SQL Central) [page 329]
Use SQL Central to start a database without connecting to it.

Starting a Database Without Connecting (SQL) [page 330]
Start a database without connecting to it by using the START DATABASE statement.

Stopping a Database After Disconnecting (SQL Central) [page 331]

Stop a database you are currently connected to by first disconnecting from the database, and then stopping it.

Stopping a Database After Disconnecting (SQL) [page 332]
Stop a database you are connected to, by disconnecting from it, and then stop it using the STOP DATABASE statement.

Related Information

Connection Parameters to Use with Embedded Databases [page 42]
Maximum Page Size Considerations [page 333]
DatabaseFile (DBF) Connection Parameter [page 68]
START DATABASE Statement

1.3.6.1 Starting a Database Without Connecting (SQL Central)

Use SQL Central to start a database without connecting to it.

Prerequisites

By default, you must have the SERVER OPERATOR system privilege. The required privileges can be changed by using the -gd database server option.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to another database.
2. In the left pane, select the database server and then click *File* > *Start Database*.
3. In the *Start Database* window, enter the required values to start a different database without connecting to it.

Results

The database appears under the database server as a disconnected database.

Related Information

Database Servers [page 318]
START DATABASE Statement

1.3.6.2 Starting a Database Without Connecting (SQL)

Start a database without connecting to it by using the START DATABASE statement.

Prerequisites

By default, you must have the SERVER OPERATOR system privilege. The required privileges can be changed by using the -gd database server option.

Procedure

1. Connect to another database.
2. Execute a START DATABASE statement.

Results

The database is started.

Example

Start the database file `c:\temp\temp.db` on the current database server.

```
START DATABASE 'c:\\temp\\temp.db'
AS tempdb
AUTOSTOP OFF;
```

The AUTOSTOP OFF connection parameter prevents the database from being stopped automatically when all connections have been disconnected.

Related Information

Database Servers [page 318]
START DATABASE Statement

1.3.6.3 Stopping a Database After Disconnecting (SQL Central)

Stop a database you are currently connected to by first disconnecting from the database, and then stopping it.

Prerequisites

By default, you must have the SERVER OPERATOR system privilege. The required privileges can be changed by using the -gd database server option.

You must be connected to another database on the same database server to stop a database.

Procedure

1. Make sure you are connected to another database on the same database server. If there is no other database running on the database server, you can connect to the utility database.
2. In SQL Central, select the database you want to stop and click *File* > *Stop Database*.

Results

When disconnecting from the database, the database may disappear from the left pane. This occurs if your connection was the only remaining connection, and if AutoStop was specified when the database was started. AutoStop causes the database to be stopped automatically when the last connection disconnects.

Related Information

How to Stop the Database Server [page 325]
Connecting to the Utility Database (Connect Window) [page 311]
AutoStop (ASTOP) Connection Parameter [page 54]
STOP DATABASE Statement

1.3.6.4 Stopping a Database After Disconnecting (SQL)

Stop a database you are connected to, by disconnecting from it, and then stop it using the STOP DATABASE statement.

Prerequisites

By default, you must have the SERVER OPERATOR system privilege. The required privileges can be changed by using the -gd database server option.

You must be connected to another database on the same database server to stop a database.

Context

Databases started with the AutoStop (ASTOP) connection parameter automatically stop when the last user disconnects.

Procedure

1. Make sure you are connected to another database on the same database server. If there is no other database running on the database server, you can connect to the utility database.
2. Execute a STOP DATABASE statement.

Results

The database is stopped.

Example

The following statements connect to the utility database and stops the tempdb database.

```
CONNECT to 'TestEng' DATABASE utility_db
AS conn2
USER 'DBA'
IDENTIFIED BY 'passwd';
STOP DATABASE tempdb;
```

Related Information

How to Stop the Database Server [page 325]
Connecting to the Utility Database (Connect Window) [page 311]
AutoStop (ASTOP) Connection Parameter [page 54]
STOP DATABASE Statement

1.3.7 Database Server Configuration

Many database server configuration features are provided.

- You can choose from many options to specify database server configuration features as how much memory to use as cache, how many CPUs to use (on multi-processor computers running a network database server), and which network protocols to use (network server only).
  Options are one of the major ways of tuning behavior and performance.
- You can run the server as a Windows service. When you run the server as a service, the server continues running even when you log off the computer.
- You can start the personal server from an application and shut it down when the application has finished with it. This configuration is typical when using the database server as an embedded database.

Related Information

How to Run the Database Server as a Service or Daemon [page 354]
Connection Parameters to Use with Embedded Databases [page 42]
How to Start the Database Server [page 324]
SQL Anywhere Database Server Executable (dbsrv17, dbeng17) [page 380]

1.3.8 Maximum Page Size Considerations

The database server cache is arranged in **pages**. Pages are fixed-size areas of memory.

Since the database server uses a single cache for its lifetime (until it is shut down), all pages must have the same size.

A database file is also arranged in pages, with a size that is specified when the database is created. Every database page must fit into a cache page. By default, the database server page size is the same as the largest page size of the databases that are specified when the database server is started. Once the database server starts, you cannot start a database with a larger page size than the database server page size.

To allow databases with larger page sizes to be started after startup, you can force the database server to start with a specific page size by using the -gp option. If you use larger page sizes, remember to increase your cache size. A cache of the same size accommodates only a fraction of the number of the larger pages, leaving less flexibility in arranging the space.

The following command starts a database server that reserves a 64 MB cache and can accommodate databases of page sizes up to 8192 bytes.

```
dbsrv17 -gp 8192 -c 64M -n myserver
```

Related Information

-gp Database Server Option [page 443]

1.3.9 Database Server Names and Database Names

You can use -n as a database server option (to name the database server) or as a database option (to name the database).

Client applications can use connection parameters that specify the database server name and database name when connecting to a database. The database server name appears on the desktop icon and in the title bar of the database server messages window.

You cannot create a database or start a database server with the name utility_db because this name is reserved for the utility database.

Naming the Database Server

Providing a database server name helps avoid conflicts with other database server names on your network. It also provides a meaningful name for users of client applications. The database server keeps its name for its lifetime (until it is shut down). If you don't provide a database server name, the database server is given the name of the first database started.

Unless you are using the Broadcast Repeater utility (dbns17), you can have different servers on different subnets with the same name.

You can name the database server by supplying a -n option before the first database file. For example, the following command starts a database server running the sample database and gives the database server the name Cambridge:

```
dbsrv17 -n Cambridge "%SQLANYSAMP17%\demo.db"
```

If you supply a database server name, you can start a database server without starting a database. The following command starts a database server named Galt with no database started:

```
dbsrv17 -n Galt
```

Database server names must be valid identifiers. Long database server names are truncated to different lengths depending on the protocol. Database server names cannot:

- begin with white space, single quotes, or double quotes

- end with white space
- contain semicolons, forward slashes (/), or backslashes (\)
- be longer than 250 bytes
- contain spaces when they are running on Unix

> **i** Note
>
> On Windows and Unix, version 9.0.2 and earlier clients cannot connect to version 10.0.0 and later database servers with names longer than the following lengths:
>
> - 40 bytes for Windows shared memory
> - 31 bytes for Unix shared memory
> - 40 bytes for TCP/IP

Naming Databases

You may want to provide a meaningful database name for users of client applications. The database is identified by that name until it is stopped. The maximum length for database names is 250 bytes.

If you don't provide a database name, the default name is the root of the database file name (the file name without the `.db` extension). For example, in the following command the first database is named mydata, and the second is named mysales.

```
dbsrv17 c:\mydata.db c:\sales\mysales.db
```

You can name databases by supplying a -n option following the database file. For example, the following command starts the sample database and names it MyDB:

```
dbsrv17 "%SQLANYSAMP17%\demo.db" -n MyDB
```

Case Sensitivity

Database server names and database names are case insensitive as long as the character set is single-byte.

Related Information

The Utility Database (utility_db) [page 309]
Database Starting and Stopping [page 328]
Connection Strings and Character Sets [page 607]

1.3.10 Configuration Files and Database Server Startup Options

You can store the set of options used to start a database server in a configuration file and invoke that file in a database server command.

The configuration file can contain options on several lines. For example, the following configuration file starts a database server and the sample database. It sets a cache of 10 MB, and names this instance of the personal server Elora. Lines with # as the first character in the line are treated as comments.

```
# Configuration file for server Elora
-n Elora
-c 10M
"C:\Users\Public\Documents\SQL Anywhere
         17\Samples\demo.db"
```

If you name the file containing these options `sample.cfg`, you could use the file as follows:

```
dbsrv17 @sample.cfg
```

Related Information

Configuration Files [page 1117]
@data Database Server Option [page 397]
Conditional Parsing in Configuration Files [page 1119]

1.3.11 Database Server Logging

The **database server message log** contains informational messages, errors, warnings, and messages from the MESSAGE statement.

These messages can appear in the following locations:

- the database server messages window (a system tray icon on Windows)
- the SQL Central *Server Messages And Executed SQL* pane
- Cockpit
- the database server message log file
- a command prompt window or shell when running the database server as a command line application
- the Unix Syslog

In this section:

Turn on logging SQL statements in SQL Central.

Stopping Logging Database Changes [page 339]
Stop logging database changes using SQL Central.

Redirecting Messages to a File Using a Facility Identifier on UNIX/Linux [page 339]
Redirect database server messages to a file that is available to the system administrators on UNIX and Linux.

Related Information

-o Database Server Option [page 471]
-oe Database Server Option [page 472]
-on Database Server Option [page 473]
-os Database Server Option [page 475]
-ot Database Server Option [page 476]

1.3.11.1 How to Log Database Server Messages to a File

By default, database server messages are sent to the database server messages window, but you can send the output to a database server message log file using the -o option.

Run the following command to output to a message log file named `mydbserver_messages.txt`:

```
dbsrv17 -o mydbserver_messages.txt -c ...
```

You can control the size of the database server message log file, and specify what you want done when a file reaches its maximum size:

- Use the -o option to specify that a database server message log file should be used and to provide a name.
- Use the -ot option to specify that a database server message log file should be used and provide a name when you want the previous contents of the file to be deleted before new messages are sent to it.
- In addition to -o or -ot, use the -on option to specify the size at which the database server message log file is renamed with the extension `.old` and a new file is started with the original name.
- In addition to -o or -ot, use the -os option to specify the size at which a new database server message log file is started with a new name based on the date and a sequential number.

You can specify a separate file where startup errors, fatal errors, and assertions are logged using the -oe option.

It is recommended that you do not end the database server message log file name with `.log` because this can create problems for utilities that perform operations using the transaction log.

Related Information

-o Database Server Option [page 471]

-oe Database Server Option [page 472]
-on Database Server Option [page 473]
-os Database Server Option [page 475]
-ot Database Server Option [page 476]

1.3.11.2 Logging SQL Statements

Turn on logging SQL statements in SQL Central.

Context

As you work with a database in SQL Central, the application automatically generates SQL statements depending on your actions.

You can keep track of these statements in a separate pane, called *Server Messages And Executed SQL*, or save the information to a file.

The *Server Messages And Executed SQL* pane has a tab for each database and database server. The tab for database servers contains the same information as the database server messages window.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. Turn on logging of SQL statements generated by SQL Central to the *Server Messages and Executed SQL* pane.
   a. Click *View* > *Server Messages and Executed SQL*.
   b. In the *Server Messages And Executed SQL* pane, click the tab with the database icon.
   c. Right-click and click *Options* in the dropdown menu.
   d. Edit the logging options.
   e. Click *Save*.
3. Turn on logging of SQL statements generated by SQL Central to a file.
   a. In the left pane, right-click the database and click *Start Logging Database Changes*.
   b. Specify a file name and click *Save*.

Results

Database changes are logged.

Related Information

Logging Statements (Interactive SQL) [page 1084]

1.3.11.3 Stopping Logging Database Changes

Stop logging database changes using SQL Central.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, right-click the database.
3. Click *Stop Logging Database Changes*.

Results

Database changes are no longer logged.

1.3.11.4 Redirecting Messages to a File Using a Facility Identifier on UNIX/Linux

Redirect database server messages to a file that is available to the system administrators on UNIX and Linux.

Context

The following steps explain how to redirect messages on Solaris, but you can also do this on other platforms such as Linux and IBM AIX.

On other platforms, such as HP-UX, the `syslog.conf` file is found in a different location. You can place the `/var/adm/sqlanywhere` file in whatever location you want.

Procedure

1. Choose a unique facility identifier that is not already being used by another application that is running on your system.

   You can do this by looking in the `/etc/syslog.conf` file to see of any of the local*n* facilities are referenced.
2. Edit the `/etc/syslog.conf` file and add the following line, where local*n* is the facility identifier you chose in step 1:

   ```
   localn.err;localn.info;localn.notice  /var/adm/sqlanywhere
   ```
3. Create the `/var/adm/sqlanywhere` file:

   ```
   touch /var/adm/sqlanywhere
   ```
4. Tell the syslog process that you have modified the `syslog.conf` file by finding the process ID of syslog:

   ```
   ps -ef | grep syslog
   ```

   and then running the following command where `pid` is the process ID of syslog:

   ```
   kill -HEAP pid
   ```
5. Start your database server with the following command, where **`localn`** is the facility identifier you chose in step 1:

   ```
   dbsrv17 -s localn ...
   ```

Results

Any messages that the database server reports to Syslog are redirected to the `/var/adm/sqlanywhere` file.

Related Information

-s Database Server Option [page 489]

1.3.12 How to Suppress Microsoft Windows Event Log Messages

You can suppress Windows event log entries by setting a registry entry.

The registry entry is `Software\SAP\SQL Anywhere\17.0`. This entry can be placed in either the HKEY_CURRENT_USER or HKEY_LOCAL_MACHINE hive.

To control event log entries, set the EventLogMask key, which is of type REG_DWORD. The value is a bit mask containing the internal bit values for the different types of event messages:

```
errors      EVENTLOG_ERROR_TYPE        0x0001
warnings    EVENTLOG_WARNING_TYPE      0x0002
information EVENTLOG_INFORMATION_TYPE  0x0004
```

When changing the setting of the EventLogMask key, you must restart the database server for the change to take effect.

Example

For example, if the EventLogMask key is set to 0, no messages appear. When you set this key to 1, informational and warning messages do not appear, but errors do. The default setting (no entry present) is for all message types to appear.

Related Information

Network Protocol Options [page 187]

1.3.13 Special Modes

You can run SQL Anywhere in special modes.

Read-only

You can run databases in read-only mode by supplying the -r option. Databases that have auditing turned on cannot be started in read-only mode.

In-memory mode

You can run databases entirely in memory by specifying the -im option. When you run in checkpoint mode only (-im c), the database server does not use a transaction log, but the database can be recovered to the most recent checkpoint. When you run the database in never write mode (-im nw), committed transactions are not written to the database file on disk, and all changes are lost when you shut down the database. Using either in-memory mode, your application can still make changes to the database or access it while the database is active.

Bulk load

This is useful when loading large quantities of data into a database using the Interactive SQL INPUT statement. Do not use the -b option if you are using LOAD TABLE to bulk load data.

Starting without a transaction log

Use the -f database option for recovery, either to force the database server to start after the transaction log has been lost, or to force the database server to start using a transaction log it would otherwise not find. The -f option is a database option, not a server option.

Once the recovery is complete, you should stop your server and restart without the -f option.

Operating quietly

The database server supports quiet mode. You determine how quiet you want the server to operate, ranging from suppressing messages or the icon in the system tray, to complete silence. To operate a completely silent database server on Windows, specify the -qi, -qs, and -qw options. With these options set, there is no visual indication that the server is running as all icons and all possible startup error messages are suppressed. If you run the database server in quiet mode, you can use either (or both) the -o or -oe options to diagnose errors.

The -qi and -qs options do not suppress windows caused by the -v (version) and -ep (prompt for database encryption password) server options.

Related Information

Data Import and Export
-r Database Server Option [page 488]
-r Database Option [page 556]
-im Database Server Option [page 454]
-b Database Server Option [page 400]
-f Database Option [page 552]
-qi Database Server Option [page 483]
-qn Database Server Option [page 484]
-qs Database Server Option [page 486]
-qw Database Server Option [page 487]

1.3.14 How to Control Performance and Memory

Several settings can affect database server performance.

Cache size

The amount of cache memory available to the database server can be a key factor in affecting performance. The more memory made available to the database server, the faster it performs. The cache holds information that may be required more than once. Accessing information in cache is faster than accessing it from disk. The default initial cache size is computed based on the amount of physical memory, the operating system, and the size of the database files. The database server automatically adjusts the cache size as necessary.

The database server messages window displays the size of the cache at startup, and you can use the following statement to obtain the current size of the cache:

```
SELECT PROPERTY( 'CurrentCacheSize' );
```

The following table summarizes the database server options available for controlling the cache.

Cache feature	Database server option	Used for
Cache size	-c	Sets the initial amount of memory for the database server cache
	-ca 0	Enforces a static cache size
	-ch	Sets the maximum cache size for automatic cache resizing
	-chx	Sets the maximum cache size for automatic cache resizing without reserving address space for non-cache use (32-bit database servers only)
	-cl	Sets the minimum cache size for automatic cache resizing
	-cs	Displays statistics about dynamic cache size changes in the database server messages window
Cache warming	-cc	Collects information about database pages that can be used for cache warming the next time the database is started
	-cr	Warms the cache with database pages
	-cv	Displays messages about cache warming in the database server messages window

Multiprogramming level

The database server's multiprogramming level specifies the maximum number of database server tasks that can execute concurrently. In general, a higher multiprogramming level increases the overall throughput of the database server by permitting more requests to execute simultaneously. However, if the requests compete for the same resources, increasing the multiprogramming level can lead to additional contention and lengthen transaction response time.

By default, the database server's multiprogramming level is automatically adjusted. In some cases you can lower the throughput of the system by increasing the multiprogramming level. The following options allow you to control the database server's multiprogramming level manually:

Database server option	sa_server_option value	Description
-gn database server option	CurrentMultiProgrammingLevel	Sets the multiprogramming level of the database server.
-gna database server option	AutoMultiProgrammingLevel	Turns on and off dynamic tuning of the database server's multiprogramming level.
-gnh database server option	MaxMultiprogrammingLevel	Sets the maximum number of tasks that the database server can execute concurrently.

Database server option	sa_server_option value	Description
-gnl database server option	MinMultiProgrammingLevel	Sets the minimum number of tasks that the database server can execute concurrently.
-gns database server option	AutoMultiProgrammingLevelStatistics	Controls whether statistics about the automatic changes to the multiprogramming level appear in the database server message log.
-gta database server option	ProcessorAffinity	Instructs the database server which logical processors to use on Windows or Linux.

Number of processors

If you are running on a multi-processor computer using a network database server, you can set the number of processors with the -gt option.

The number of CPUs that the database server can use may also be affected by your license or SQL Anywhere edition.

Other performance-related options

There are several options available for tuning network performance, including -gb (database process priority), and -u (buffered disk I/O).

Related Information

Database Server Startup Options [page 390]
Dynamic Cache Sizing [page 1443]
Database Server Configuration of the Multiprogramming Level [page 348]
Threading [page 344]
Editions and Licensing
SQL Anywhere Database Server Executable (dbsrv17, dbeng17) [page 380]
sa_server_option System Procedure

1.3.15 Threading

To understand the threading model, you must understand the basic terminology and concepts of threading and request processing.

Request

A request is a unit of work, such as a query or SQL statement, that is sent to the database server over a connection. The lifetime of a request spans the time from when the request is first received by the database server to the time that the last of the results are returned, cursors are closed, or the request is canceled.

Task

A task is a unit of activity that is performed within the database server, and is the smallest unit of work that is scheduled by the database server. Within the database server, each user request becomes at least one task, and possibly more if intra-query parallelism is involved. In addition to user requests, the database server can also schedule its own tasks to perform internal tasks, such as running the database cleaner or processing timers. The maximum number of active tasks that can execute concurrently depends on the size of the worker pool within the kernel. If a user request task arrives at the database server and a worker is assigned to it, the ActiveReq server property is incremented by 1. Once the request completes, the ActiveReq server property is decremented by 1. If, however, there were no available workers to execute the user request task, then the task is queued for execution and the UnschReq server property is incremented by 1. When the task is dequeued for execution because a worker is now available, the UnschReq server property is decremented by 1, and the ActiveReq property is incremented by 1.

Worker

A worker is an abstraction of an executing **thread-of-control** within the database server kernel. On Microsoft Windows and Linux, workers are implemented using lightweight threads called **fibers**, and on other platforms workers are implemented using operating system threads. Workers execute tasks that are assigned to them by the database server kernel. The database server utilizes a variable-sized pool of workers to handle the server's workload. The size of the pool corresponds to the database server's multiprogramming level.

Personal servers have a fixed-sized pool. For these servers, the number of workers created at server startup is controlled by the -gn option. For all other network servers, the database server creates the number of workers specified by the -gnh option. However, only the number of workers specified by the server multiprogramming level are allowed to service tasks. By default the size of the worker pool is dynamically tuned by the kernel in response to changes in the database server's workload, and can fluctuate between the lower and upper bounds specified by the -gnl and -gnh options respectively.

Tasks are assigned to workers on a first-in, first-out (FIFO) basis. Each task's priority is set based on the connection that generated that task. Once a task is assigned to a worker for execution, the kernel's scheduler takes the task's priority into account when allocating CPU time to that worker. During task execution, if a task needs to block for some reason during its processing, such as while waiting for a lock or for I/O to complete, the worker remains coupled to that task. When the task completes, only then does the worker become available to execute other tasks.

Thread

A thread, or thread of execution, is an operating system construct that permits concurrent execution within a single process. Every operating system process, including the database server, is executed by at least one, and possibly many threads. A thread is scheduled outside the application by the operating system, and ultimately, all of an application's execution is performed by its threads. On Microsoft Windows and Linux, the database server creates a fixed number of threads: one operating system thread per CPU core, controlled by the -gtc option. On other platforms, the number of operating system threads created is equivalent to the size of the worker pool and is controlled using the same mechanisms that are used to control the worker pool size.

The number of threads is independent of the number of connections to the database, and the database server does not dedicate a thread to a specific connection. Instead, as tasks enter the database server for execution, they are dynamically assigned a thread from a fixed-size pool of server threads. Once a task is assigned to a thread, the thread processes the task until the task completes or is canceled.

In this section:

Workers on UNIX/Linux [page 346]

On UNIX and Linux, a worker is implemented using an operating system thread.

Workers on Microsoft Windows and Linux [page 346]
On Microsoft Windows and Linux, workers are implemented using lightweight threads known as **fibers**.

Threading Behavior [page 347]
There are several factors that control threading behavior, each of which are governed by a server option. Not all of these options are supported on every platform.

Database Server Configuration of the Multiprogramming Level [page 348]
The database server **multiprogramming level** is the maximum number of tasks that can be active at a time.

Related Information

Transaction Blocking and Deadlock
-gn Database Server Option [page 437]
-gnl Database Server Option [page 440]
-gnh Database Server Option [page 439]
-gtc Database Server Option [page 449]
sa_clean_database System Procedure

1.3.15.1 Workers on UNIX/Linux

On UNIX and Linux, a worker is implemented using an operating system thread.

Scheduling of tasks is therefore controlled by the operating system scheduler and the operating system may choose to preempt the execution of a thread at any time. This preemptive scheduling does not affect the processing of tasks in any visible way; when a thread is scheduled to run again, the task is picked up at the point where it left off. However, task priorities that are inherited from a database connection will have no effect on the execution scheduling of those tasks. Rather, all tasks will be executed at the same priority level because all operating system thread priorities are equivalent.

1.3.15.2 Workers on Microsoft Windows and Linux

On Microsoft Windows and Linux, workers are implemented using lightweight threads known as **fibers**.

An implementation utilizing fibers allows workers to be scheduled between themselves co-operatively, rather than being scheduled preemptively by the operating system thread scheduler. The result is that a context switch between fibers can be more finely controlled. The database kernel schedules the fibers to maximize processor affinity and tightly control the execution priority of each worker.

Because fibers do not rely on the operating system scheduler, a fiber must explicitly yield control to another fiber when it is waiting for some other activity to complete. For example, if a task that is executing on a fiber needs to block while waiting for an I/O operation to complete, it relinquishes control to another fiber. The

thread hosting the original fiber is free to pick up another fiber immediately and begin its execution without a kernel context switch. If a fiber blocks and does not yield control, it blocks the thread that is hosting it and prevents other fibers from running on that thread. If more than one thread is hosting fibers, only the thread that is hosting the waiting fiber is blocked: other threads are still free to run fibers.

On Microsoft Windows and Linux platforms there are at least as many fibers created as required by the maximum concurrency setting of the network database server, as specified by the -gnh option. The network database server utilizes a subset of these fibers during server execution, corresponding to the current server multiprogramming level. For each fiber the operating system must reserve address space for its stack. The amount of stack required for each fiber depends on the setting of the -gss server option.

The amount of address space required by a network server for execution stacks is proportional to the maximum multiprogramming level.

Related Information

-gss Database Server Option [page 445]
-gn Database Server Option [page 437]

1.3.15.3 Threading Behavior

There are several factors that control threading behavior, each of which are governed by a server option. Not all of these options are supported on every platform.

Multiprogramming level (-gn server option)

The -gn option controls the database server's multiprogramming level. This value determines the maximum number of tasks that may be active at one time. Each database request causes the creation of at least one task, and possibly more if intra-query parallelism is involved. Additionally, the server occasionally schedules tasks to perform internal activities. When the number of tasks in the server exceeds the multiprogramming level, outstanding tasks must wait until a currently running, or active task, completes.

Stack size per internal execution thread (-gss server option)

You can set the stack size per worker in the database server using the -gss option. The -gss option allow you to lower the address space requirements of each worker within the database server, which may be useful in environments with limited memory.

Number of processors (-gt server option)

The -gt option controls the number of processors that the database server uses.

Processor concurrency (-gtc server option)

The -gtc option specifies the number of logical processors (cores) that the database server uses.

Topology-aware scheduling (-gtp server option)

The -gtp option specifies that tasks are scheduled to use a single thread on each available core before attempting to use a second thread on any core.

Processor use and threading example

The following example explains how the database server selects CPUs based on the settings of -gt and -gtc. For the following examples, assume you have a system with 4 processors, with 2 cores on each processor. The physical processors are identified with letters, and the cores with numbers, so this system has processing units A0, A1, B0, B1, C0, C1, D0, and D1.

Scenario	Network database server settings
A single CPU license or -gt 1 specified	• -gt 1 • -gtc 2 Threads can execute on A0 and A1.
No licensing restrictions on the CPU with -gtc 5 specified	• -gt 4 • -gtc 5 Threads can execute on A0, A1, B0, C0, and D0.
A database server with a 3 CPU license and -gtc 5 specified	• -gt 3 • -gtc 5 Threads can execute on A0, A1, B0, B1, and C0.
No licensing restrictions on the CPU with -gtc 1 specified	• -gt 4 • -gtc 1 Threads can execute only on A0.

Related Information

Database Server Configuration of the Multiprogramming Level [page 348]
-gn Database Server Option [page 437]
-gss Database Server Option [page 445]
-gt Database Server Option [page 446]
-gtc Database Server Option [page 449]

1.3.15.4 Database Server Configuration of the Multiprogramming Level

The database server **multiprogramming level** is the maximum number of tasks that can be active at a time.

When a client-side request arrives at the database server, the task created for that request is assigned to a worker, if one is available. A request with a worker assigned to it is called an **active request**. If all available workers are busy, then the request is placed in a special queue called the unscheduled request queue and the request is classified as an Unscheduled Request. Similarly, an active task is one that is currently being serviced by a worker. An active task may be executing an access plan operator, or performing some other function, but

may also be blocked, waiting for a resource (such as an I/O operation, or a lock on a row). An unscheduled task is one that is ready to execute, but is waiting for an available worker. The number of active tasks that can execute simultaneously depends on the number of server threads and the number of logical processors in use on the computer.

SQL Anywhere lets DBAs choose between allowing the database server to dynamically tune the multiprogramming level based on server throughput (the default) or configuring the multiprogramming level manually. You can configure the multiprogramming level settings when you start a database server by specifying network database server options (-gna, -gnl, and -gnh), or after the database server is running by using the MinMultiprogrammingLevel, MaxMultiprogrammingLevel, and CurrentMultiprogrammingLevel properties with the sa_server_option system procedure.

The following table summarizes the command line and server options that control the database server's multiprogramming level:

Database server option (starting database servers)	sa_server_option value (running database servers)	Description
-gn	CurrentMultiProgrammingLevel	Sets the multiprogramming level of the database server.
-gna	AutoMultiProgrammingLevel	Turns on and off dynamic tuning of the network database server's multiprogramming level
-gnh	MaxMultiprogrammingLevel	Sets the maximum number of tasks that the network database server can execute concurrently
-gnl	MinMultiProgrammingLevel	Sets the minimum number of tasks that the network database server can execute concurrently

Tuning the Multiprogramming Level

SQL Anywhere network servers support both dynamic and manual tuning of the server's multiprogramming level.

Dynamic tuning of the multiprogramming level

SQL Anywhere network servers can automatically monitor the throughput level of the database server and determine how the multiprogramming level should be adjusted in response to the current workload. The network database server uses a hill-climbing algorithm, as well as a parabola approximation approach, to decide on the adjustment that needs to be made. If an increase in the multiprogramming level results in an increase in the network database server throughput level, then the network database server proceeds with the increase. If the increase in the multiprogramming level results in degradation in throughput, then the network database server lowers the multiprogramming level. The network database server continuously monitors the throughput level and changes the multiprogramming level to improve server throughput. For workloads that consist of short bursts of a large number of requests followed by long idle periods, it is best to set the minimum multiprogramming level to the maximum expected concurrency level. This configuration ensures that the network database server is responsive during the short bursts of requests.

Dynamic tuning of the multiprogramming level is enabled by default (-gna 1). The -gnl and -gnh network database server options set the minimum and maximum values for the multiprogramming level that the

dynamic tuning algorithm uses. When dynamic tuning is turned on, the network database server also attempts to eliminate thread deadlock issues by automatically increasing the multiprogramming level each time a thread deadlock condition arises. The network database server continues increasing the multiprogramming level up to the allowable MaxMultiprogrammingLevel value. Once the MaxMultiprogrammingLevel value is reached, the network database server starts returning thread deadlock issues to client applications.

Manual tuning of the multiprogramming level

You must turn off dynamic tuning of the multiprogramming level before you can adjust it manually. It is recommended that you test your application's workload to analyze the effects of the database server's multiprogramming level on server throughput and request response times. You can use the Microsoft Windows Performance Monitor or the Performance Statistics utility (dbstats) on UNIX and Linux to help you analyze database server behavior when testing your application. Performance statistics can be queried using the PROPERTY, DB_PROPERTY, and CONNECTION PROPERTY functions.

The performance counters related to active and unscheduled requests should be observed during this analysis.

If the number of active requests is always less than the value of the -gn database server option, you can consider lowering the multiprogramming level, but you must take into account the effects of intra-query parallelism, which adds additional tasks to the database server's execution queues. If the effect of intra-query parallelism is marginal, lowering the multiprogramming level can be done safely without reducing overall system throughput. However, if the number of total requests (active + unscheduled) is often larger than the value specified by -gn, then an increase in the multiprogramming level may be warranted. You should consider the performance tradeoffs of increasing the multiprogramming level before changing it.

In this section:

Related Information

-gn Database Server Option [page 437]
-gna Database Server Option [page 438]
-gnh Database Server Option [page 439]
-gnl Database Server Option [page 440]
sa_server_option System Procedure

1.3.15.4.1 Effects of Different Multiprogramming Level Settings

Multiprogramming level settings can impact performance.

It can be difficult to determine the optimal value for the multiprogramming level. For example, if a database application uses Java stored procedures, or if intra-query parallelism is enabled, then the additional server tasks that are created to process these requests may exceed the current multiprogramming level, and execution of these tasks must wait until another request completes. In these cases, raising the multiprogramming level may be appropriate. You can adjust the multiprogramming level of a network database server at any time. However, for personal database servers you cannot change the multiprogramming level after the server has started.

Raising the Multiprogramming Level

Often, increases to the multiprogramming level correspondingly increase the database server's overall throughput because an increase to the multiprogramming level allows additional tasks to execute concurrently. However, there are tradeoffs in raising the multiprogramming level that should be considered. They include the following:

Increased contention

By increasing the number of concurrent tasks, you may increase the probability of contention between active requests. The contention can involve resources such as schema or row locks, or on data structures and/or synchronization primitives internal to the database server. Such a situation may actually decrease server throughput.

Additional server overhead

Each active task requires the allocation and maintenance of a worker and additional bookkeeping structures to control its scheduling. In addition, each worker requires the preallocation of address space for its execution stack. The size of the stack varies by platform. On Microsoft Windows operating systems, the allocation of stack space affects the address space of the database server process, but the stack memory is allocated on demand. On UNIX and Linux platforms, the backing memory for the stack is allocated immediately. As a result, setting a higher multiprogramming level increases the server's memory footprint, and reduces the amount of memory available for the cache manager because the amount of available address space is reduced.

Thrashing

The database server can reach a state when it uses significant resources simply to manage its execution overhead, rather than doing useful work for a specific request. This state is commonly called thrashing. Thrashing can occur, for example, when too many active tasks are competing for space in the database server cache, but the cache is not large enough to accommodate the working set of database pages used by the set of active tasks. This situation can result in page stealing, in a manner similar to that which can occur with operating systems.

Impact on query processing

The database server selects a maximum number of memory-intensive requests that can be processed concurrently. Even if you increase the database server's multiprogramming level, requests may need to wait for memory to become available.

Memory for data structures

The database server uses resources to parse and optimize statements. For very complex statements or small cache sizes, the memory consumed for server data structures can exceed the amount that is available. A memory governor limits the amount of memory used for each task's server data structures. Each task has the following approximate limit:

```
(3/4 maximum-cache-size) / (number-of-active-requests)
```

If this limit is exceeded, the statement fails with an error.

Lowering the Multiprogramming Level

Reducing the database server's multiprogramming level by lowering the number of concurrently executing tasks usually lowers the database server's throughput. However, lowering the multiprogramming level may improve the response time of individual requests because there are fewer requests to compete for resources, and there is a lower probability of lock contention.

When the multiprogramming level is set too low, thread deadlock can occur. If the multiprogramming level is at a reasonable level for the workload, the occurrence of thread deadlock is symptomatic of an application design problem that results in substantial contention, and as a result, impairs scalability. One example is a table that every application must modify when inserting new data to the database. This technique is often used as part of a scheme to generate primary keys. However, the result is that it effectively serializes all the application's insert transactions. When the rate of insert transactions becomes higher than what the server can service because of the serialization on the shared table, thread deadlock usually occurs.

The -gnl database server option sets the minimum multiprogramming level.

Related Information

Cache and the Memory Governor [page 1441]
Deadlocks
Windows Performance Monitor [page 1506]
Connection, Database, and Database Server Properties [page 879]
-gn Database Server Option [page 437]
-gna Database Server Option [page 438]
-gnh Database Server Option [page 439]
-gnl Database Server Option [page 440]
-gns Database Server Option [page 441]
sa_server_option System Procedure
Performance Statistics Utility (dbstats) (UNIX/Linux) [page 1193]

1.3.15.4.2 Manual Adjustment of the Multiprogramming Level

You can adjust a network server multiprogramming level manually.

However, if auto tuning of the multiprogramming level is enabled, then the server may re-adjust the multiprogramming level automatically sometime after the manual setting. If you lower the multiprogramming level, the new setting may not take effect immediately, as the server may have to wait for active tasks to complete before the new multiprogramming level can take effect. If you attempt to set the multiprogramming level to a value outside the lower and upper bounds as specified by the MinMultiProgrammingLevel and MaxMultiProgrammingLevel settings, then an error will be returned.

You can disable automatic multiprogramming level tuning either when you start the database server or when the database server is running:

Disabling automatic multiprogramming tuning at server startup

When you start the network database server, specify -gna 0 to disable automatic tuning of the multiprogramming level:

```
dbsrv17 -gna 0 ...
```

You can change the default multiprogramming level using the -gn option, and set the minimum and maximum values using the -gnl and -gnh options, respectively.

Disabling automatic multiprogramming tuning of a running server

If the network database server is already running, execute the following SQL statement to disable automatic tuning of the multiprogramming level:

```
CALL sa_server_option ( 'AutoMultiProgrammingLevel', 'NO' );
```

You can set the minimum and maximum values using the MinMultiProgrammingLevel and MaxMultiProgrammingLevel properties, respectively. For example:

```
CALL sa_server_option ( 'MinMultiProgrammingLevel', 10 );
CALL sa_server_option ( 'MaxMultiProgrammingLevel', 100 );
```

You can set the current multiprogramming level using the CurrentMultiProgrammingLevel property as follows:

```
CALL sa_server_option ( 'CurrentMultiProgrammingLevel', 25 );
```

Related Information

-gna Database Server Option [page 438]
-gn Database Server Option [page 437]
-gnh Database Server Option [page 439]
-gnl Database Server Option [page 440]
sa_server_option System Procedure

1.3.16 How to Run the Database Server as a Service or Daemon

There are several ways in which the database server can be run as a service or daemon.

When you log on to a computer using a user ID and a password, you establish a **session**. When you start a database server, or any other application, it runs within that session. When you log off the computer, all applications associated with the session shut down.

If you need the database server to be available all the time, you can run SQL Anywhere for Microsoft Windows and for UNIX and Linux so that when you log off the computer, the database server remains running.

Microsoft Windows service

You can run the database server on Microsoft Windows as a service. This configuration can be convenient for running high availability servers.

UNIX/Linux daemon

You can run the database server on UNIX and Linux as a daemon using the -ud option, which enables the database server to run in the background and continue running after you log off.

Linux service

You can run the database server on Linux as a service. This configuration has many convenient properties for running high availability servers.

Run the dbsvc utility to see a complete list of the programs that can be run as services.

In this section:

Check and change which group a service belongs to in SQL Central.

Database Server as a Daemon on UNIX/Linux [page 368]
To run the database server in the background on UNIX or Linux, and to enable it to run independently of the current session, you run it as a **daemon**.

Related Information

Service Utility (dbsvc) for Linux [page 1207]
Service Utility (dbsvc) for Windows [page 1212]

1.3.16.1 Windows Services

You can run the database server like a Microsoft Windows program rather than a service.

However, there are limitations to running it as a standard program and especially in multi-user environments.

Limitations of Running as a Standard Executable

When you start a program, it runs under your Windows login session, which means that if you log off the computer, the program shuts down. This configuration restricts the use of the computer to keep a program running most of the time, as is commonly the case with database servers. You must stay logged on to the computer running the database server for the database server to keep running. This configuration can also present a security risk as the Windows computer must be left in a logged on state.

Advantages of Services

Installing an application as a Windows service enables it to run even when you log off.

When you start a service, it logs on using a special system account called LocalSystem (or another account that you specify). Since the service is not tied to the user ID of the person starting it, the service remains open even when the person who started it logs off. You can also configure a service to start automatically when the Windows computer starts, before a user logs on.

Managing Services

SQL Central provides a more convenient and comprehensive way of managing SQL Anywhere services than the Windows services manager. You can also use the dbsvc utility to create and modify services.

Related Information

Service Utility (dbsvc) for Windows [page 1212]

1.3.16.2 Creating Windows Services (SQL Central)

Create a Windows service using SQL Central.

Context

Each database server can run more than one database. When you run more than one database at a time, it is recommended that you do so by adding new databases to your existing Windows service, rather than by creating new services.

Procedure

1. In SQL Central, connect to the database.
2. In the left pane, click *SQL Anywhere 17*.
3. In the right pane, click the *Services* tab.
4. Click *File* > *New* > *Service*.
5. Follow the instructions in the *Create Service Wizard*.
   - Service names must be unique within the first eight characters.
   - When creating a service in SQL Central, type options for the executable, without the executable name itself, in the window. For example, if you want a network server to run using the sample database with a cache size of 20 MB and the name myserver, you would type the following in the *Parameters* text box of the *Create Service Wizard* in SQL Central:

     ```
     -c 20M
     -n myserver "%SQLANYSAMP17%\demo.db"
     ```

     Line breaks are optional.
   - Choose the account under which the service will run: the special LocalSystem account or another user ID.
   - If you choose to start a service automatically, it starts whenever the computer starts Windows. If you choose to start the service manually, you must start the service from SQL Central each time. You may want to click *Disabled* if you are setting up a service for future use.

Results

A new Windows service is created.

Related Information

Service Account Options [page 362]
MobiLink Server Use Outside the Current Session
Configuring Windows Services [page 360]
Service Utility (dbsvc) for Windows [page 1212]

1.3.16.3 Creating Windows Services (dbsvc Utility)

Create a Windows service using the dbsvc utility.

Context

Each database server can run more than one database. When you run more than one database at a time, it is recommended that you do so by adding new databases to your existing Windows service, rather than by creating new services.

Procedure

Run the Service utility (dbsvc) with the -w option.

Results

A new Windows service is created.

Example

The following command creates a Windows service called myserv where the database server runs as the LocalSystem user.

```
dbsvc -as -w myserv "C:\Program Files\SQL Anywhere 17\Bin32\dbsrv17.exe"
```

```
-n william -c 8m "c:\temp\sample.db"
```

Related Information

MobiLink Server Use Outside the Current Session
Configuring Windows Services [page 360]
Service Utility (dbsvc) for Windows [page 1212]

1.3.16.4 Deleting Windows Services (SQL Central)

Delete a Windows service using SQL Central, which removes the service name from the list of services.

Prerequisites

The service must be stopped.

Context

Deleting a Windows service does not remove any software from your hard disk.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *SQL Anywhere 17*.
3. In the right pane, click the *Services* tab.
4. Right-click the service you want to remove, and then click *Delete*.

Results

The Windows service is deleted.

Related Information

Service Utility (dbsvc) for Windows [page 1212]

1.3.16.5 Deleting Windows Services (dbsvc Utility)

Delete a Windows service using the dbsvc utility, which removes the service name from the list of services.

Prerequisites

The service must be stopped.

Context

Deleting a Windows service does not remove any software from your hard disk.

Procedure

Run the Service utility (dbsvc) with the -d option.

Results

The Windows service is deleted.

Example

Run the following command to delete the service called myserv, without prompting for confirmation.

```
dbsvc -y -d myserv
```

Related Information

Service Utility (dbsvc) for Windows [page 1212]

1.3.16.6 Configuring Windows Services

Configure a Windows service.

Context

A Windows service runs database servers or other applications with a set of options and properties, including the properties that specify the account under which the service runs and the conditions under which it starts.

Changes to a service configuration take effect the next time someone starts the service.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *SQL Anywhere 17*.
3. In the right pane, click the *Services* tab.
4. Right-click the service you want to configure and click *Properties*.
5. Alter the properties as needed on the tabs of the *Service Properties* window.
6. Click *OK* when finished.

Results

The properties for the service are altered. The *Startup* property is applied the next time Windows is started.

In this section:

Windows Service Startup Options [page 361]
Windows service startup options govern startup behavior for SQL Anywhere services.

Windows Service Options [page 361]
The Widows service options for a service are the same as the options for the executable.

Service Account Options [page 362]
You can choose which account the service runs under.

Changing the Service Executable File [page 362]
Change the program executable file associated with a service.

Adding a Database to a Windows Service [page 363]
Add a database to a Windows service.

1.3.16.6.1 Windows Service Startup Options

Windows service startup options govern startup behavior for SQL Anywhere services.

You can set them on the *General* tab of the *Service Properties* window.

Automatic

The service starts whenever the Windows operating system starts. This setting is appropriate for database servers and other applications that run all the time.

Manual

In Windows, you must be an administrator to start the service.

Disabled

The service does not start.

1.3.16.6.2 Windows Service Options

The Widows service options for a service are the same as the options for the executable.

The *Configuration* tab of the *Service Properties* window provides a *Parameters* text box for specifying options for a service. Do not type the name of the program executable in this box.

Example

To start a network server service with a cache size of 20 MB, named my_server and running two databases, you would type the following in the *Parameters* field:

```
-c 20M
-n my_server
c:\db_1.db
c:\db_2.db
```

Here's another network server example that changes the database names that you connect to:

```
-c 20M
-n my_server
c:\databases\db_1.db -n mydb1
c:\databases\db_2.db -n mydb2
```

In this example, an application connection string could specify a parameter like DBN=mydb1 or DatabaseName=mydb2 to indicate which database to choose.

To start a SQL Remote Message Agent service that connects to the sample database as user DBA, the following connection option could be used:

```
-c "UID=DBA;PWD=sql;DBN=demo"
```

1.3.16.6.3 Service Account Options

You can choose which account the service runs under.

By default, most services run under the special LocalSystem account. You can set the service to log on under another account by opening the *Log On* tab on the service *Properties* window, and typing the account information.

If you choose to run the service under an account other than LocalSystem, that account must have the *Log On As A Service* privilege. This privilege can be granted from the Windows User Manager application under Advanced Privileges. The Service utility (dbsvc) can also be used to grant this privilege (using the -a option).

When an Icon Appears in the System Tray

- If a service runs under LocalSystem, and *Allow Service To Interact With Desktop* is selected on the *Log On* tab, an icon appears in the system tray of every user logged in to Windows on the computer running the service. Any user can open the application window and stop the program running as a service.
- If a service runs under LocalSystem, and *Allow Service To Interact With Desktop* is cleared on the *Log On* tab, no icon appears in the system tray for any user. Only users with permissions to change the state of services can stop the service.
- If a service runs under another account, no icon appears in the system tray. Only users with permissions to change the state of services can stop the service.

> **i** Note
>
> Services cannot directly interact with a user as of Windows Vista and Windows Server 2008. The *Allow Service To Interact With Desktop* option cannot change this.

1.3.16.6.4 Changing the Service Executable File

Change the program executable file associated with a service.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.

2. In the left pane, click *SQL Anywhere 17*.
3. In the right pane, click the *Services* tab.
4. Right-click the service you want to configure and click *Properties*.
5. Click the *Configuration* tab
6. In the *File name* text box, type the new path and file name

   If you move an executable file to a new directory, you must modify this entry.

Results

The program executable is updated.

1.3.16.6.5 Adding a Database to a Windows Service

Add a database to a Windows service.

Context

Each database server can run more than one database. When you run more than one database at a time, it is recommended that you do so by adding databases to your existing Windows service, rather than by creating new services.

Databases can be started on running servers by client applications, such as Interactive SQL.

Starting a database from an application does not attach it to the service. If the service is stopped and restarted, the additional database does not start automatically.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *SQL Anywhere 17*.
3. In the right pane, click the *Services* tab.
4. Right-click the service and click *Properties*.
5. Click the *Configuration* tab.
6. Add the path and file name of the new database to the end of the list of options in the *Parameters* box.
7. Click *OK* to save the changes.

Results

The database starts the next time the Windows service starts.

Related Information

START DATABASE Statement
db_start_database Function

1.3.16.7 Setting the Refresh Frequency

Set the refresh frequency in SQL Central.

Context

By default, SQL Central refreshes the information it displays every 10 seconds. Refreshing checks the status (started or stopped) of each Windows service, and updates the *Status* column to display the current state. In addition, SQL Central refreshes the dynamic objects such as connections and locks, as well as the dynamic properties of events and maintenance plans.

Procedure

1. In SQL Central, click *Tools* > *SQL Anywhere 17* > *Preferences*.
2. Click the *Automatic Refresh* tab.
3. Click *Enable Automatic Refreshing*.
4. Specify a time interval in the *Refresh Every* x *seconds*.
5. Click *Apply*.
6. Click *OK*.

Results

The refresh frequency is altered.

1.3.16.8 Starting or Stopping a Windows Service (SQL Central)

Start or stop a Windows service in SQL Central.

Context

If you start a service, it keeps running until you stop it. Closing SQL Central or logging off does not stop the service.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *SQL Anywhere 17*.
3. In the right pane, click the *Services* tab.
4. Right-click the service, and then click *Start* or *Stop*.

Results

The Windows service is started or stopped. Stopping a database server service closes all connections to the database and stops the database server. For other applications, the program shuts down.

1.3.16.9 The Windows Service Manager

The Windows Service Manager organizes tasks that can be performed on your computer.

You can use SQL Central to perform all the service management for SQL Anywhere. Although you can use the Windows *Service Manager* in the *Control Panel* for some tasks, you cannot install or configure a SQL Anywhere service from the Windows *Service Manager*.

If you open the Windows *Service Manager* (from the Windows *Control Panel*), a list of services appears. The names of the SQL Anywhere services are formed from the service name you provided when installing the service, prefixed by SQL Anywhere. All the installed services appear together in the list.

1.3.16.10 Managing Service Groups

Check and change which group a service belongs to in SQL Central.

Prerequisites

You must ensure that your service is a member of an appropriate group before you can configure your services to start in the correct order.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *SQL Anywhere 17*.
3. In the right pane, click the *Services* tab.
4. Right-click the service and click *Properties*.
5. Click the *Dependencies* tab. The top text box displays the name of the group the service belongs to.
6. Click *Change* to display a list of available groups on your system.
7. Select one of the groups, or type a name for a new group.
8. Click *OK* to assign the service to that group.

Results

The group services are checked and changed.

In this section:

Related Information

List of Service Groups [page 367]

1.3.16.10.1 Managing Service Dependencies

Add a service or a group to a list of dependencies using SQL Central.

Context

In some circumstances you may want to run more than one executable as a service, and these executables may depend on each other. For example, you may want to run a server and a SQL Remote Message Agent to assist in replication. Services must start in the correct order. If a SQL Remote Message Agent service starts before the server has started, it fails because it cannot find the server. You can prevent these problems by using service groups, which you manage from SQL Central. With SQL Central you can specify dependencies for a service. For example:

- You can ensure that at least one member of each of a list of service groups has started before the current service.
- You can ensure that any number of services have started before the current service. For example, you may want to ensure that a particular network server has started before a SQL Remote Message Agent that is to run against that server starts.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *SQL Anywhere 17*.
3. In the right pane, click the *Services* tab.
4. Right-click the service and click *Properties*.
5. Click the *Dependencies* tab.
6. Click *Add Services* or *Add Service Groups* to add a service or group to the list of dependencies.
7. Select one of the services or groups from the list.
8. Click *OK* to add the service or group to the list of dependencies.

Results

The service or group is added to a list of dependencies.

1.3.16.10.2 List of Service Groups

Lists the service groups that a service or daemon can belong to.

By default, each service belongs to a group, as listed in the following table.

Service	Default group
Network server	SQLANYServer
Personal server	SQLANYEngine
MobiLink synchronization client	SQLANYMLSync
SQL Remote Message Agent	SQLANYRemote
MobiLink server	SQLANYMobiLink
SQL Anywhere Broadcast Repeater utility	SQLANYNS
MobiLink Listener	SQLANYLSN
SQL Anywhere Volume Shadow Copy Service	SQLANYVSS
MobiLink Relay Server	SQLANYRSHOST (Linux only)
MobiLink Relay Server Outbound Enabler	SQLANYRSOE
MobiLink Agent	SQLANYMLAGENT

1.3.16.11 Database Server as a Daemon on UNIX/Linux

To run the database server in the background on UNIX or Linux, and to enable it to run independently of the current session, you run it as a **daemon**.

> **i** Note
>
> Do not use '&' to run the database server in the background. If you use the UNIX/Linux & (ampersand) command to run the database server in the background, it does not work. The database server either immediately shuts down or stops responding. You must instead run the database server as a daemon.
>
> As well, attempting to start a server in the background from within a program using the typical `fork()`–`exec()` sequence does not work. If you need to do this, add the -ud option to the list of database server options.

You can run the UNIX/Linux database server as a daemon in one of the following ways:

1. Use the -ud option when starting the database server. For example:

   ```
   dbsrv17 -ud demo
   ```

2. Use the dbspawn tool to start the database server. For example:

   ```
   dbspawn dbsrv17 demo
   ```

   One advantage of using dbspawn is that the dbspawn process does not shut down until it has confirmed that the daemon has started and is ready to accept requests. If for any reason the daemon fails to start, the exit code for dbspawn is non-zero.
   When you start the daemon directly using the -ud option, the database server creates the daemon process and returns immediately (exiting and allowing the next command to be executed) before the daemon initializes itself or attempts to open any of the databases specified in the command.

To ensure that a daemon is running one or more applications that can use the database server, you can use dbspawn to ensure that the daemon is running before starting the applications. The following is an example of how to test this using a csh script.

```
#!/bin/csh
# start the server as a daemon and ensure that it is
# running before you start any applications
dbspawn dbsrv17 demo
if ( $status != 0 ) then
   echo Failed to start demo server
   exit
endif
# ok, now you can start the applications
...
```

This example uses an sh script to test whether the daemon is running before starting the applications.

```
#!/bin/sh
# start the server as a daemon and ensure that it is
# running before you start any applications
dbspawn dbsrv17 demo
if [ $? != 0 ]; then
   echo Failed to start demo server
   exit
fi
# ok, now you can start the applications
...
```

3. Spawn a daemon from within a C program. The -ud option must be used. For example:

```
...
if( fork() == 0 ) {
      /* child process = start server daemon */
      execl( "/opt/sqlanywhere17/bin/dbsrv17",
"dbsrv17", "-ud", "demo" );
   exit(1);
}
/* parent process */
...
```

Related Information

-ud Database Server Option [page 512]
Start Server in Background Utility (dbspawn) [page 1217]

1.3.17 Authenticated Applications for OEM Editions

The OEM Edition of SQL Anywhere can perform any operation on the database.

The OEM Edition of SQL Anywhere is provided for SAP OEM Partners. With the OEM Edition of SQL Anywhere, an authenticated application can perform any operation on the database, subject to the privileges granted to the user ID.

Unauthenticated connections have read-only access, and can perform inserts, updates, and deletes on temporary tables. Using unauthenticated connections allows complex reports to be created using stored procedures, and accessed using reporting tools, such as Crystal Reports.

The authentication mechanism is independent of any application programming language or tool, and is carried out on every connection, so you can use both authenticated connections and more restricted unauthenticated connections in your application.

Authentication is not a security mechanism. Anyone running an unauthenticated database server against the database can perform any operation, subject to the usual SQL privileges scheme.

In this section:

1.3.17.1 Authenticated Application Development

Developing an authenticated application is a simple process: a special authentication signature is incorporated into the database, and a second signature is incorporated into your application.

When the application connects to the database, the signatures are compared to authenticate the application. The following steps are required to develop an authenticated SQL Anywhere application:

1. Obtain an authentication signature
2. Authenticate the database
3. Authenticate the application

All the database tools included with SQL Anywhere, including SQL Central, Interactive SQL, and the utilities, such as dbbackup, are self-authenticating. They are unrestricted in their operations against any authenticated database. If the database itself is not authenticated, the tools act in a restricted, read-only fashion.

You must use the OEM Edition of the SQL Anywhere database server for an authenticated application. This edition differs from the usual database server only in that it processes authentication instructions. The authentication instructions are ignored by other editions of the database server. If you do not use the authenticated database server, no restrictions are placed on unauthenticated applications.

In this section:

You can execute the SET TEMPORARY OPTION statement that sets the authentication option depending on the programming interface you are using.

Upgrading Authenticated Databases [page 377]
Upgrade an authenticated database by creating the `authenticate.sql` file in the *%SQLANY17%*`/scripts` directory before upgrading or rebuilding the database.

Discovering the oem_string Option Setting from an Application [page 378]
Query the oem_string option from an application.

1.3.17.1.1 Obtaining Authentication Signatures

Obtain authentication signatures for your database and application to use the OEM Edition.

Prerequisites

To get an authentication signature, you must have an OEM contract with SAP.

Procedure

1. Go to Requesting SAP SQL Anywhere Authentication Signatures.
2. Complete the form to get your authentication signatures. The following information is incorporated into your authentication mechanism:

   Company

   The name of your company.

   Application Name

   The name of your application.

Results

Once you complete the form, you are emailed a database signature and an application signature within 48 hours. These signatures are long strings of characters and digits.

Related Information

Authenticating a Database (SQL) [page 372]

1.3.17.1.2 Authenticating a Database (SQL)

Authenticate a database using the SET OPTION statement.

Prerequisites

You must have the SET ANY SECURITY OPTION system privilege to set this option.

Procedure

1. Set the database_authentication option for the database by using the following SQL authentication statement:

   ```
   SET OPTION PUBLIC.database_authentication='company=company-name;
        application=application-name;
        signature=database-signature';
   ```

   > **i Note**
   >
   > Line breaks have been added to the syntax example to improve readability. However, the syntax should be executed without line breaks and without spaces between the equal sign and semicolons.

   The `company-name` and `application-name` arguments are the values you supplied to SAP when obtaining your signature, and `database-signature` is the database signature that you received from SAP.
2. Restart the database for the option to take effect.

Results

The database is authenticated.

Next Steps

You can store the authentication statement in a SQL script file to avoid having to type in the long signature. You can run the SQL script from Interactive SQL by clicking *File* > *Run Script*.

If you create a file named `authenticate.sql` in the `scripts` subdirectory of your SQL Anywhere installation directory and store the authentication statement in this file, it is applied whenever you create, rebuild, or upgrade a database.

Related Information

Upgrading Authenticated Databases [page 377]
SET OPTION Statement
database_authentication Option [page 715]
List of Database Server Properties [page 900]

1.3.17.1.3 Checking That a Database Is Authenticated

Check that your database is authenticated using the DB_PROPERTY function.

Context

When the database server loads an authenticated database, it displays a message in the *Database server messages* window describing the authenticated company and application. The message has the following form:

```
This database is licensed for use with:
  Application: application-name
  Company: company-name
```

Procedure

1. Connect to the database.
2. Execute the DB_PROPERTY function, as follows:

```
SELECT DB_PROPERTY ( 'Authenticated' );
```

Results

A message is returned indicating whether the database is authenticated.

Related Information

DB_PROPERTY Function [System]
List of Database Server Properties [page 900]
database_authentication Option [page 715]

1.3.17.1.4 Authenticating Your Application

An authenticated application must set the connection_authentication database option immediately after connecting.

The option must be set on every connection immediately after the connection is established. ODBC or JDBC applications query the database about its capabilities, and you may not have control over these actions. For this reason, every connection has a thirty second grace period before the restrictions apply. The grace period allows an application to authenticate regardless of which development tool is being used.

You can use the Authenticated database property to determine if the database has activated authentication:

```
SELECT DB_PROPERTY( 'Authenticated' );
```

You can use the Authenticated connection property to determine if the current connection has been authenticated:

```
SELECT CONNECTION_PROPERTY ( 'Authenticated' );
```

The following SQL statement authenticates the connection.

```
SET TEMPORARY OPTION connection_authentication='company=company-name;
     application=application-name;
     signature=application-signature';
```

> **i Note**
>
> Line breaks have been added to the syntax example to improve readability. However, the syntax should be executed without line breaks and without spaces between the equal sign and semicolons.

The option must be set for the duration of the connection only by using the TEMPORARY keyword. The `company-name` and `application-name` must match the values specified by the database_authentication option. The `application-signature` is the signature that you obtained from SAP.

The database server verifies the application signature against the database signature. If the signature is verified, the connection is authenticated and has no restrictions on its activities beyond those imposed by the user's assigned roles and privileges. If the signature is not verified, the connection is limited to those actions permitted by unauthenticated applications.

Related Information

List of Connection Properties [page 880]
connection_authentication Option [page 707]

1.3.17.1.5 Authentication Statement Execution

You can execute the SET TEMPORARY OPTION statement that sets the authentication option depending on the programming interface you are using.

The signatures listed here are not valid signatures. Examples are provided for setting the authentication option using the following interfaces:

- Microsoft ADO.NET
- Embedded SQL
- JDBC
- ODBC
- OLE DB
- SAP PowerBuilder

> **i Note**
>
> If your company name has quotation marks, apostrophes, or other special characters (for example, Joe's Garage) you need to be careful about how you construct the authentication statement. The entire set of authentication options (Company=...;Application=...;Signature=...) is a SQL string. The rules for strings in SQL dictate that if you include a quotation mark inside the string, it must be doubled to be accepted. For example:
>
> ```
> SET TEMPORARY OPTION connection_authentication='Company = Joe''s Garage;
>     Application=Joe''s Program;
>     Signature=0fa55157edb8e14d818e...';
> ```
>
> Line breaks have been added to the syntax example to improve readability. However, the syntax should be executed without line breaks and without spaces between the equal sign and semicolons.

Microsoft ADO.NET

Use the following statement:

```
SACommand cmd=new SACommand(
   "SET TEMPORARY OPTION connection_authentication=
       'Company=MyCo;
        Application=MyApp;
        Signature=0fa551599998e14d818e...';",
    con
);
cmd.ExecuteNonQuery();
```

The string must be entered on a single line, or you must build it up by concatenation.

Embedded SQL

Use the following statement:

```
EXEC SQL SET TEMPORARY OPTION connection_authentication=
   'Company=MyCo;
    Application=MyApp;
    Signature=0fa551599998e14d818e...';
```

The string must be entered on a single line, or you must build it up by concatenation.

JDBC

Use the following statement:

```
Statement Stmt1 = con.createStatement();
Stmt1.executeUpdate(
    "SET TEMPORARY OPTION connection_authentication=
    'Company=MyCo;
     Application=MyApp;
     Signature=0fa55159999e14d818e...';"
);
```

The string must be entered on a single line, or you must build it up by concatenation.

ODBC

Use the following statement:

```
SQLExecDirect(
    hstmt,
    "SET TEMPORARY OPTION connection_authentication=
       'Company=MyCo;
        Application=MyApp;
        Signature=0fa55159999e14d818e...';",
    SQL_NTS
);
```

The string must be entered on a single line, or you must build it up by concatenation.

OLE DB

When connecting to an authenticated database, the connection and authentication steps are performed separately. However, some objects, such as the Microsoft Visual Basic Grid object can attempt a separate, implicit connection, which does not automatically include authentication. In such cases, the connection is not authenticated and the database operation can fail. You can avoid this problem by including the InitString connection parameter in the connection string. The following example illustrates how you can modify a

Microsoft Visual Basic application to include the InitString connection parameter so that every connection is immediately followed by authentication:

```
mConnectionString =
   "Provider=SAOLEDB.17;
    UID=DBA;
    PWD=passwd;
    ServerName=test17;
    InitString=SET TEMPORARY OPTION connection_authentication=
       'Company=MyCo;
        Application=MyApp;
        Signature=0fa55157edb8e14d818e...'"
mdbConn.ConnectionString = mConnectionString
mdbConn.Open
```

The string must be entered on a single line, or you must build it up by concatenation.

SAP PowerBuilder

Use the following PowerScript statement:

```
EXECUTE IMMEDIATE
   "SET TEMPORARY OPTION connection_authentication=
       'Company=MyCo;
        Application=MyApp;
        Signature=0fa551599998e14d818e...';"
USING SQLCA
```

1.3.17.1.6 Upgrading Authenticated Databases

Upgrade an authenticated database by creating the `authenticate.sql` file in the *%SQLANY17%*`/scripts` directory before upgrading or rebuilding the database.

Prerequisites

To set PUBLIC options, you must have the SET ANY PUBLIC OPTION system privilege

It is recommended that you make backup copies of your database files.

Context

The file is run by the CREATE DATABASE and ALTER DATABASE UPGRADE statements, dbupgrad and dbunload utilities, and the *Upgrade Database Wizard*.

The only way to preserve authentication information when upgrading or rebuilding a database is to store the authentication statement in the file `authenticate.sql`.

Procedure

1. Create a file named `authenticate.sql` in the *%SQLANY17%*`\scripts` directory.
2. Add the following contents to the file:

   ```
   SET OPTION PUBLIC.database_authentication = 'authentication-statement'
   go
   ```

   The `go` must appear in the file; otherwise, the statement is ignored.

Results

The database is upgraded.

Next Steps

Upgrade or rebuild the database. The steps you must follow depend on the version of the database file you are upgrading or rebuilding.

Related Information

The Rebuild Process for Version 9 and Earlier Databases
The Upgrade Process for Version 10 and Later Databases
The Rebuild Process for Version 10 and Later Databases
database_authentication Option [page 715]

1.3.17.1.7 Discovering the oem_string Option Setting from an Application

Query the oem_string option from an application.

Context

The oem_string database option stores user-specified information in the header page of a database file.

The oem_string option is set for authenticated applications.

Procedure

1. C and execute a SET OPTION statement to set the database OEM string and then query the setting.

```
SET OPTION PUBLIC.oem_string='Jack''s OEM Database';
SELECT CONNECTION_PROPERTY( 'oem_string' );
```

2. Disconnect from the database and stop the database (for example, by shutting down the database server).
3. Open the database system dbspace file (the .db file) using a hexadecimal editor or some user-defined application.
4. Read the first page of the file into a buffer.
5. Search the buffer for the two-byte prefix (DB_OEM_STRING_PREFIX) and suffix (DB_OEM_STRING_SUFFIX) sequences before and after the OEM string. The following is an example.

```
0220: 00 00 00 00  00 00 00 00  00 00 00 00  DA 7A BA 5E |.............z.^|
0230: 00 45 4D 3D  4A 61 63 6B  27 73 20 4F  45 4D 20 44 |.EM=Jack's OEM D|
0240: 61 74 61 62  61 73 65 00  00 00 00 00  00 00 00 00 |atabase.........|
0250: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00 |................|
0260: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00 |................|
0270: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00 |................|
0280: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00 |................|
0290: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00 |................|
02a0: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00 |................|
02b0: 00 00 00 00  3D 4D 45 00  5E BA 7A DA  5E 77 CB A4 |....=ME.^.z.^w..|
```

Results

The prefix and suffix strings are defined in `sqldef.h` as DB_OEM_STRING_PREFIX and DB_OEM_STRING_SUFFIX, respectively. All the bytes between these two strings define the OEM string that is defined in the database.

Related Information

oem_string Option [page 780]

1.3.18 SQL Anywhere on Windows

There are several considerations that apply when running SQL Anywhere software on Windows.

Security

These operating systems incorporate a security model called User Account Control (UAC). UAC is enabled by default and may affect the behavior of programs that expect to be able to write files, especially when the computer supports more than one user. Depending on where and how files and directories are created, a file created by one user may have permissions that do not allow another user to read or write to that file. If

you install SQL Anywhere into the default directories, then files and directories that require read/write access for multiple users are set up appropriately.

SQL Anywhere elevated operations agent

Certain actions require privilege elevation to execute when run under UAC. The following programs may require elevation in SQL Anywhere:

- `dbdsn.exe`
- `dbelevate17.exe`
- `dblic.exe`
- `dbsvc.exe`
- `installULNet.exe`
- `SetupVSPackage.exe`
- `ulcond17.exe`

The following DLLs require elevation when they are registered or unregistered:

- `dbctrs17.dll`
- `dbodbc17.dll`
- `dboledb17.dll`
- `dboledba17.dll`

If UAC is activated, you may receive an elevation prompt for the SQL Anywhere elevated operations agent. The prompt is issued by the User Account Control system to confirm that you want to continue running the identified program (if logged on as an administrator) or to provide administrator credentials (if logged on as a non-administrator).

Deployment considerations

The program `dbelevate17.exe` is used internally by SQL Anywhere components to perform operations that require elevated privileges. This executable must be included in deployments of SQL Anywhere.

SQL Anywhere executables signed

SQL Anywhere executables are signed by SAP.

Windows services

Services that are compliant with Windows are not allowed to interact with the desktop. On these operating systems, no SQL Anywhere services interact with the desktop. You can monitor database servers using SQL Central or SQL Anywhere Cockpit.

1.4 SQL Anywhere Database Server Executable (dbsrv17, dbeng17)

Starts and runs the database server.

Syntax

```
dbsrv17 [ server-options ] [ database-file [ database-options ] ...]
```

```
dbeng17 [ server-options ] [ database-file [ database-options ] ...]
```

Server Options

Server option	Description
@ data	Reads in options from a configuration file or environment variable.
-?	Displays usage information.
-al userid [; userid ...]	Allow standard user authentication for specified users.
-b	Runs in bulk operations mode.
-c size	Sets initial cache size.
-ca 0	Disables dynamic cache sizing.
-cc{ + | - }	Collects information about database pages to be used for cache warming.
-cdb option [;option ...]	Starts the Cockpit.
-ch size	Sets the cache size upper limit.
-chx size	Reserves address space for non-cache use.
-cl size	Sets the cache size lower limit.
-cp location [; location ...]	Specifies set of directories or JAR files in which to search for classes.
-cr { + | - }	Warms the cache with database pages.
-cs	Displays cache usage in the database server messages window.
-cv{ + | - }	Controls the appearance of messages about cache warming in the database server messages window.
-dt temp-file-dir	Specifies the directory where temporary files are stored.
-ec encryption-options	Enables packet encryption.
-edi	Turns on database isolation.
-ep	Prompts for encryption key.

Server option	Description
-es	Allows unencrypted connections over shared memory.
-f	Forces the database to start without a transaction log.
-fc `filename`	Specifies the file name of a DLL containing the file system full callback function.
-fips	Requires the use of FIPS-certified algorithms for database and communication encryption.
-ga	Automatically unloads the database after the last non-HTTP client connection is closed. In addition, shuts down after the last database is closed.
-gb `level`	Sets database process priority class to `level`.
-gc `num`	Sets maximum checkpoint timeout period to `num` minutes.
-gd `level`	Sets database starting privilege.
-ge `size`	Sets the stack size for threads that run external functions.
-gf	Disables firing of triggers.
-gk `level`	Sets the privilege required to stop the server.
-gl `level`	Sets the privilege required to load or unload data.
-gm `num`	Sets the maximum number of connections.
-gn `num`	Sets the multiprogramming level of the database server.
-gna	Controls automatic tuning of the database server multiprogramming level.
-gnh `num`	Sets the maximum number of tasks that the database server can execute concurrently.
-gnl `num`	Sets the minimum number of tasks that the database server can execute concurrently.
-gns	Reports multiprogramming level statistics in the database server message log.
-gp `size`	Sets the maximum page size to `size` bytes.
-gr `minutes`	Sets the maximum recovery time.
-gss `size`	Sets the thread stack size to `size` bytes.

Server option	Description
-gt `num`	Sets the maximum number of physical processors that can be used (up to the licensed maximum). This option is only useful on multiprocessor systems.
-gta `logical-processors-to-use`,...	Sets which logical processors the database server can use.
-gtc `logical-processors-to-use`	Controls the maximum processor concurrency that the database server allows.
-gtp { *0* | *1* }	Sets topology-aware scheduling off or on.
-gu `level`	Sets the privilege level for utility commands: utility_db, all, none, or DBA.
-im `submode`	Runs the database server in memory, reducing or eliminating writes to disk.
-k	Controls the collection of Performance Monitor statistics.
-kl `GSS-API-library-file`	Specifies the file name of the Kerberos GSS-API library (or shared object on UNIX and Linux) and enable Kerberos authenticated connections to the database server.
-kp `server-principal`	Specifies the Kerberos server principal and enable Kerberos authenticated connections to the database server.
-kr `server-realm` (deprecated)	Specifies the realm of the Kerberos server principal and enables Kerberos authenticated connections to the database server.
-krb	Enables Kerberos-authenticated connections to the database server.
-ks	Disables the creation of shared memory that the Performance Monitor uses to collect counter values from the database server.
-ksc	Specifies the maximum number of connections that the Performance Monitor can monitor.
-ksd	Specifies the maximum number of databases that the Performance Monitor can monitor.
-lt `n`	Specifies the number of lock tables.
-m	Truncates the transaction log after each checkpoint for all databases.
-md	Sets the crash dump type. Default is NORMAL.

Server option	Description
-n `name`	Uses `name` as the name of the database server. The -n option is positional.
-ncs	Enables sending status events to the SAP Solution Manager using the NCS (Native Component Supportability) library.
-ncsd `filename`	Enables sending status events to the SAP Solution Manager using the NCS (Native Component Supportability) library, and specifies the location of the NCS configuration file to use.
-o `filename`	Outputs messages to the specified file.
-oe `filename`	Specifies file to log startup errors, fatal errors, and assertions to.
-on `size`	Specifies a maximum size for the database server message log file, after which the file is renamed with the extension `.old` and a new file is started.
-os `size`	Limits the size of the message log file.
-ot `filename`	Truncates the database server message log file and appends output messages to it.
-p `packet-size`	Sets the maximum communication packet size.
-pc	Compresses all communication packets except same-computer connections.
-pf `filename`	Writes the process ID into the specified file.
-phl { *OFF* | *ON* | `property-name`, ... }	Sets history tracking for specified database server properties.
-phs { [`HH`:]`MM`:`SS` | `size` [*K* | *M*] | *max* | *default* }	Sets the maximum amount of memory to use for tracking property history, in allotted time or bytes.
-pt `size-in-bytes`	Sets the minimum network packet size to compress.
-qi	Does not display the database server system tray icon or database server messages window.
-qn	Does not minimize the database server messages window on startup.
-qp	Suppresses messages about performance in the database server messages window.
-qs	Suppresses startup error windows.

Server option	Description
-qw	Does not display the database server messages window.
-r	Opens database in read-only mode.
-s `facility-ID`	Sets the Syslog facility ID.
-sb { *0* | *1* }	Specifies how the server reacts to broadcasts.
-sbx { + | - }	Controls the default disk sandbox settings for all databases started on the database server that do not have explicit disk sandbox settings.
-sf `feature-list`	Secures features for databases running on this database server.
-sjvm { *ON* | *OFF* }	Specifies that the database server uses one Java VM for all databases running on the database server.
-sk `key`	Specifies a key that can be used to enable features that are disabled for the database server.
-su `password`	Sets the password for the DBA user of the utility database (utility_db), or disable connections to the utility database.
-tdsl	Sets the TDS login request mode.
-ti `minutes`	Sets the client idle time before shutdown (default 240 minutes).
-tl `seconds`	Sets the default liveness timeout for clients in seconds (default 120 seconds).
-tmf	Forces transaction manager recovery for distributed transactions.
-tmt `milliseconds`	Sets the re-enlistment timeout for distributed transactions.
-tq `time`	Sets quitting time.
-ts	Sets up a database server trace session.
-u	Uses buffered disk I/O.
-ua	Turns off use of asynchronous I/O.
-uc	Starts the database server in shell mode.
-ud	Runs as a daemon.
-uf	Specifies the action that the database server takes when a fatal error or assertion failure occurs on a database server.

Server option	Description
-ufd	Specifies the action that the database server takes when a fatal error or assertion failure occurs on a database.
-ui	Opens the *Server Startup Options* window and displays the database server messages window, or starts the database server in shell mode if a usable display isn't available.
-um	Opens the *Server Startup Options* window and displays the database server messages window if `DBLauncher.app` is running.
-uq	Starts the database server in shell mode but suppresses output.
-ut `minutes`	Touches temporary files every `min` minutes.
-ux	Displays the database server messages window and *Server Startup Options* window.
-v	Displays database server version and stop.
-vss{ + | - }	Enables and disables the Volume Shadow Copy Service (VSS).
-wc[+ | -]	Enables write checksums for databases running on the database server.
-x `list`	Specifies a comma-separated list of communication protocols to use.
-xa `authentication-info`	Specifies a list of database names and authentication strings for an arbiter server.
-xd	Prevents the database server from becoming the default database server.
-xf `state-file`	Specifies the location of the file used for maintaining state information about your database mirroring system.
-xm `seconds`	Sets the time to check for new IP addresses in seconds. The minimum value is 10 and the default value is 0. For a portable device, the default value is 120.
-xs	Specifies server-side web services communications protocols.
-z	Provides diagnostic information about communication links.
-ze	Displays database server environment variables in the database server messages window.

Server option	Description
-zl	Turns on capturing of the most recently prepared SQL statement for each connection.
-zn `integer`	Specifies the number of request log file copies to retain.
-zo `filename`	Redirects request logging information to a separate file.
-zoc `filename`	Redirects web service client information to a file.
-zp	Turns on capturing of the plan most recently used by the query optimizer.
-zr `level`	Turns on logging of SQL operations. The default is NONE.
-zs `size`	Limits the size of the request log file.
-zt	Turns on logging of request timing information.

Database Options

The following options can only be specified after a database file name in the database server command.

Database option	Description
-a `filename`	Applies the named transaction log file.
-ad `log-directory-specification`	Specifies the directory (or directories) containing transaction log files to be applied to the database.
-al `userid` [*;* `userid` ...]	Allow standard user authentication for specified users of the specified database.
-ar	Applies any log files located in the same directory as the transaction log to the database.
-as	Continues running the database after transaction logs have been applied (used in conjunction with -ad or -ar).
-dh	Does not display the database when dblocate is used against this server.
-ds	Specifies the location of the dbspaces for the database.
-ek `key`	Specifies encryption key.
-m	Truncates (deletes) the transaction log after each checkpoint for the specified database.

Database option	Description
-n `name`	Names the database. The -n option is positional.
-r	Opens the specified database(s) in read-only mode. Database modifications not allowed.
-ru	Specifies a time stamp to restore the database to during a point-in-time recovery.
-ruo	Specifies an offset in the transaction log to restore the database to during a point-in-time recovery.
-sbx { + | - }	Controls disk sandboxing for the database, which restricts read-write file operations on the database to the directory where the main database file is located and any subdirectories of this directory.
-sn `alternate-server-name`	Provides an alternate server name for a single database running on a database server.
-wc[+ | -]	Enables write checksums for databases running on the database server.
-xp on	Enables database mirroring for a partner or a copy node.

Remarks

The elements of the database server command include the following:

Executable

The dbeng17 command starts a personal database server.

The dbsrv17 command starts a network database server.

The support for TCP/IP in the network server enables you to perform tasks from your desktop computer.

On Windows, the name of the personal database server executable is *dbeng17*`.exe`. On UNIX and Linux operating systems, the name is dbeng17.

On Windows, the name of the network database server executable is *dbsrv17*`.exe`. On UNIX and Linux operating systems, the name is dbsrv17.

Server options

These options control the behavior of the database server for all running databases.

Database file

You can specify zero, one, or more database file names. Each of these databases starts and remains available for applications.

> ⚠ Caution
>
> The database file and the transaction log file must be located on the same physical computer as the database server or accessed via a SAN or iSCSI configuration. Database files and transaction log files located on a remote network directory can lead to poor performance, data corruption, and database server instability.
>
> For best results, the transaction log should be kept on a different disk from the database files.

Database options

For each database file you start, you can provide database options that control certain aspects of its behavior.

Database and server options are generally case-sensitive. You should enter all options in lowercase.

The `database-file` specifies the database file name. If `database-file` is specified without a file extension, database server looks for `database-file` with extension `.db`. If you use a relative path, it is read relative to the current working directory. You can supply a full path.

If you supply no options and no database file, then on Windows operating systems a window appears, allowing you to browse to your database file.

To start a database server from a batch file, use the dbspawn utility.

The personal database server has a maximum of ten concurrent connections, uses at most four cores on one CPU for request processing, and doesn't support network client/server connections. By default, the personal database server only uses the shared memory protocol. You must use the -x option to use TCP/IP with the personal database server.

In addition, there are other minor differences, such as the default privilege level that is required to start new databases, or the privileges required to execute the CHECKPOINT statement.

By default, the database server page size is the same as the largest page size of the databases on the command line. Once the database server starts, you cannot start a database with a larger page size than the database server.

Example

The following command starts the SQL Anywhere sample database running on a personal database server:

```
dbeng17 "%SQLANYSAMP17%\demo.db"
```

The following command starts the SQL Anywhere sample database running on a network database server:

```
dbsrv17 "%SQLANYSAMP17%\demo.db"
```

The following example starts a database server named **`myserver`** that starts with a cache size of 3 MB and loads the sample database:

```
dbsrv17 -c 3m -n myserver "%SQLANYSAMP17%\demo.db"
```

In this section:

Database Server Startup Options [page 390]
The database server options apply to the database server as a whole, not just to an individual database.

Database Startup Options [page 543]
The database options are specified on the command line after the database file, and apply only to that database.

Troubleshooting: How Database Servers Are Located [page 564]
When the client successfully locates a server, it then tries to locate the database.

Related Information

The Transaction Log [page 292]
Database Servers [page 318]
Maximum Page Size Considerations [page 333]
How to Start the Database Server [page 324]
Configuration Files [page 1117]
Running a SQL Anywhere Database File that is Stored Remotely from the Server Machine
Start Server in Background Utility (dbspawn) [page 1217]

1.4.1 Database Server Startup Options

The database server options apply to the database server as a whole, not just to an individual database.

In this section:

@data Database Server Option [page 397]
Reads in options from the specified environment variable or configuration file.

-? Database Server Option [page 398]
Displays usage information.

-al Database Server Option [page 399]
Allows standard user authentication for specified users.

-b Database Server Option [page 400]
Uses bulk operation mode.

-c Database Server Option [page 400]
Sets the initial memory reserved for caching database pages and other database server information.

-ca Database Server Option [page 403]
Enforces a static cache size.

-cc Database Server Option [page 404]
Collects information about database pages to be used for cache warming the next time the database is started.

-cdb Database Server Option [page 405]
Starts the Cockpit.

-ch Database Server Option [page 407]
Sets a maximum cache size, as a limit to automatic cache growth.

-chx Database Server Option [page 409]
Sets a maximum cache size as a limit to automatic cache growth.

-cl Database Server Option [page 410]
Sets a minimum cache size as a lower limit to dynamic cache resizing.

-cp Database Server Option [page 412]
Specifies a set of directories or JAR files in which to search for classes.

-cr Database Server Option [page 413]
Reloads (warms) the cache with database pages using information collected the last time the database was run.

-cs Database Server Option [page 414]
Displays statistics related to dynamic cache sizing in the database server messages window.

-cv Database Server Option [page 415]
Controls the appearance of messages about cache warming in the database server messages window.

-dt Database Server Option [page 416]
Specifies the directory where temporary files are stored.

-ec Database Server Option [page 418]
Uses transport layer security (TLS) or simple obfuscation to encode communication protocol packets (such as DBLIB and ODBC) transmitted to and from all clients.

-edi Database Server Option [page 422]
Turns on database isolation.

-ep Database Server Option [page 423]
Prompts the user for the encryption key upon starting a strongly encrypted database.

-es Database Server Option [page 424]
Allows unencrypted connections over shared memory.

-fc Database Server Option [page 425]
Specifies the file name of a DLL (or shared object on Unix) containing the File System Full callback function.

-fips Database Server Option [page 426]
Sets the database server to use FIPS-certified encryption for strong database and communication encryption.

-ga Database Server Option [page 427]
Unloads the database after the last non-HTTP client connection disconnects.

-gb Database Server Option [page 428]
Sets the server process priority class.

-gc Database Server Option [page 429]
Sets the maximum interval between checkpoints.

-gd Database Server Option [page 430]
Sets the privileges required to start or stop a database on a running database server.

-ge Database Server Option [page 432]
Sets the stack size for external functions.

-gf Database Server Option [page 432]
Disables firing of triggers by the server.

-gk Database Server Option [page 433]
Sets the privileges required to stop the database server.

-gl Database Server Option [page 434]
Sets the privilege required to load data using the LOAD TABLE statement, and to unload data using the UNLOAD or UNLOAD TABLE statements.

-gm Database Server Option [page 436]
Limits the number of concurrent connections to the database server.

-gn Database Server Option [page 437]
Sets the multiprogramming level of the database server.

-gna Database Server Option [page 438]
Controls automatic tuning of the database server multiprogramming level.

-gnh Database Server Option [page 439]
Sets the maximum multiprogramming level - the maximum number of tasks that the network database server can execute concurrently.

-gnl Database Server Option [page 440]
Sets the minimum database server multiprogramming level.

-gns Database Server Option [page 441]
Displays statistics for automatic multiprogramming level adjustments in the database server message log.

-gp Database Server Option [page 443]
Sets the maximum allowed database page size.

-gr Database Server Option [page 444]
Sets the maximum length of time (in minutes) for recovery from system failure.

-gss Database Server Option [page 445]
Sets the stack size per worker in the database server.

-gt Database Server Option [page 446]
Sets the maximum number of physical processors that can be used (up to the licensed maximum). This option is only useful on multiprocessor systems.

-gta Database Server Option [page 447]
Sets which logical processors the database server can use.

-gtc Database Server Option [page 449]
Controls the maximum processor concurrency that the database server allows.

-gtp Database Server Option [page 450]
Turns topology-aware scheduling on or off. This option is only useful on multiprocessor systems.

-gu Database Server Option [page 452]
Sets the privilege required for executing database file administration statements such as for creating or dropping databases.

-im Database Server Option [page 454]
Runs the database server in memory, reducing or eliminating writes to disk. With the exception of the validation option, in-memory mode requires a separate license.

Resets the password of the specified user.

-os Database Server Option [page 475]
Specifies a maximum size for the database server message log file, at which point the file is renamed.

-ot Database Server Option [page 476]
Truncates the database server message log file and appends output messages to it.

-p Database Server Option [page 477]
Sets the maximum size of communication packets.

-pc Database Server Option [page 478]
Compresses all connections except for same-computer connections.

-pf Database Server Option [page 479]
Writes the process ID into the specified file.

-phl Database Server Option [page 480]
Sets history tracking for specified database server properties.

-phs Database Server Option [page 481]
Sets the maximum amount of memory to use for tracking property history, in allotted time or bytes.

-pt Database Server Option [page 482]
Increases or decreases the size limit at which TCP/IP packets are compressed.

-qi Database Server Option [page 483]
Controls whether database server system tray icon and database server messages window appear.

-qn Database Server Option [page 484]
Specifies that the database server messages window is not minimized on startup.

-qp Database Server Option [page 485]
Specifies that messages about performance do not appear in the database server messages window.

-qs Database Server Option [page 486]
Suppresses startup error windows.

-qw Database Server Option [page 487]
Specifies that the database server messages window does not appear.

-r Database Server Option [page 488]
Forces all databases that start on the database server to be read-only.

-s Database Server Option [page 489]
Sets the user ID for Syslog messages.

-sb Database Server Option [page 490]
Specifies how the database server reacts to broadcasts.

-sbx Database Server Option [page 491]
Sets the default disk sandbox settings for all databases started on the database server that do not have explicit disk sandbox settings.

-sclr Database Server Option [page 492]
Specifies that the database server uses one CLR external environment for all databases running on the database server

-sf Database Server Option [page 493]
Controls whether users have access to features for databases running on the current database server.

-sjvm Database Server Option [page 499]

Specifies that the database server uses one Java VM for all databases running on the database server.

-sk Database Server Option [page 500]
Creates the SYSTEM secured feature key and sets the authorization code for it. This permits access to features that are secured for the database server.

-su Database Server Option [page 502]
Sets the user ID and password for the utility database (utility_db), or disables connections to the utility database.

-tdsl Database Server Option [page 503]
Sets the TDS login mode.

-ti Database Server Option [page 505]
Disconnects inactive connections.

-tl Database Server Option [page 506]
Sets the period at which to send liveness packets.

-tmf Database Server Option [page 507]
Forces transaction manager recovery for distributed transactions.

-tmt Database Server Option [page 507]
Sets a re-enlistment timeout for participation in distributed transactions.

-tq Database Server Option [page 508]
Shuts down the server at a specified time.

-ts Database Server Option [page 509]
Sets up a database server tracing session.

-u Database Server Option [page 510]
Opens files using the operating system disk cache.

-ua Database Server Option [page 511]
Turns off use of asynchronous I/O.

-uc Database Server Option [page 512]
Starts the database server in shell mode.

-ud Database Server Option [page 512]
Runs the database server as a daemon.

-uf Database Server Option [page 513]
Specifies the action that the database server takes when a fatal error or assertion failure occurs on a database server.

-ufd Database Server Option [page 514]
Specifies the action that the database server takes when a fatal error or assertion failure occurs on a database.

-ui Database Server Option [page 516]
Opens the *Server Startup Options* window, displays the database server messages window, and starts the database server whether the X window server starts or not.

-um Database Server Option [page 517]
Displays database server messages in a new window within `DBLauncher.app`.

-uq Database Server Option [page 518]
Starts the database server in shell mode but suppresses output.

-ut Database Server Option [page 518]
Touches temporary files.

-ux Database Server Option [page 519]
Opens the *Server Startup Options* window or displays the database server messages window on Linux (use the X window server).

-v Database Server Option [page 520]
Displays the software version.

-vss Database Server Option [page 521]
Enables and disables the use of the Volume Shadow Copy Service (VSS).

-wc Database Server Option [page 522]
Controls whether checksums are enabled on write operations for all databases on this database server, if the databases do not have checksums enabled by default.

-x Database Server Option [page 523]
Specifies server-side network communications protocols.

-xa Database Server Option [page 525]
Specifies a comma-separated list of database names and authentication strings for an arbiter server.

-xd Database Server Option [page 526]
Prevents the database server from becoming the default database server.

-xf Database Server Option [page 527]
Specifies the location of the file used for maintaining state information about your database mirroring system.

-xm Database Server Option [page 528]
Specifies how often the database server checks for new IP addresses.

-xs Database Server Option [page 529]
Specifies server-side web services communications protocols.

-z Database Server Option [page 532]
Displays diagnostic communication messages, and other messages, for troubleshooting purposes.

-ze Database Server Option [page 533]
Displays database server environment variables in the database server messages window.

-zl Database Server Option [page 534]
Turns on capturing of the most recently prepared SQL statement for each connection to databases on the server.

-zn Database Server Option [page 535]
Specifies the number of request log file copies to retain.

-zo Database Server Option [page 536]
Redirects request logging information to a file separate from the regular log file.

-zoc Database Server Option [page 537]
Redirects web service client information to a file.

-zp Database Server Option [page 538]
Turns on capturing of the plan most recently used by the query optimizer.

-zr Database Server Option [page 538]
Enables request logging of operations.

-zs Database Server Option [page 540]
Limits the size of the request log.

-zt Database Server Option [page 542]
Turns on logging of request timing information.

Related Information

sa_server_option System Procedure
List of Database Server Properties [page 900]

1.4.1.1 @data Database Server Option

Reads in options from the specified environment variable or configuration file.

Syntax

```
dbsrv17 @data ...
```

Applies to

All operating systems.

This option is also supported for all database utilities except the Language Selection utility (dblang) and the Start Server in Background utility (dbspawn).

Remarks

Use this option to read in command-line options from the specified environment variable or configuration file. If both exist with the same name that is specified, the environment variable is used.

Configuration files can contain line breaks, and can contain any set of options.

To protect the information in a configuration file (for example, because it contains passwords), use the File Hiding (dbfhide) utility to encrypt the contents of configuration files.

The @data parameter can occur at any point in the command, and parameters contained in the file are inserted at that point. Multiple files can be specified, and the file specifier can be used with command line options.

Example

The following example starts a database server named **myserver** that loads the sample databases. The database server does not automatically adjust to the static cache size, which is 40% of the total physical memory:

```
-c 4096
-n myserver
"c:\mydatabase.db"
```

If this configuration file is saved as `c:\config.txt`, it can be used in a command as follows:

```
dbsrv17 @c:\config.txt
```

The following configuration file contains comments:

```
#This is the server name:
-n MyServer
#These are the protocols:
-x tcpip
#This is the database file
my.db
```

The following statement sets an environment variable that holds options for a database server that starts with a cache size of 4 MB and loads a database called mydatabase.db.

```
SET envvar=-c 4096 "c:\mydatabase.db";
```

The following command starts the database server using an environment variable named **envvar**.

```
dbsrv17 @envvar
```

Related Information

Configuration Files [page 1117]
File Hiding Utility (dbfhide) [page 1160]

1.4.1.2 -? Database Server Option

Displays usage information.

Syntax

```
dbsrv17 -?
```

Applies to

All operating systems.

Remarks

When you specify this option, a brief description of each server option appears.

1.4.1.3 -al Database Server Option

Allows standard user authentication for specified users.

Syntax

```
dbsrv17 -al userid[ ;userid ... ] ...
```

On Unix, quotation marks are required when more than one parameter is supplied:

Applies to

All operating systems.

Remarks

Allow the users specified in the semicolon separated list of user IDs to use standard authentication. At most 5 user IDs can be specified.

This option is useful if the login_mode option does not include standard authentication and the user cannot authenticate using alternate means. Users with full DBA privileges can always authenticate using standard authentication.

Related Information

login_mode Option [page 752]

1.4.1.4 -b Database Server Option

Uses bulk operation mode.

Syntax

```
dbsrv17 -b ...
```

Applies to

All operating systems.

Remarks

This option is useful for using the Interactive SQL INPUT statement to load large quantities of data into a database.

The -b option should not be used if you are using LOAD TABLE to bulk load data.

When you use this option, the database server allows only one connection by one application. It keeps a rollback log, but it doesn't keep a transaction log. The multi-user locking mechanism is turned off.

When you first start the database server after loading data with the -b option, you should use a new transaction log file.

Bulk operation mode doesn't disable the firing of triggers.

Related Information

Data Recovery Issues for Bulk Operations
Performance Aspects of Bulk Operations
INPUT Statement [Interactive SQL]
LOAD TABLE Statement

1.4.1.5 -c Database Server Option

Sets the initial memory reserved for caching database pages and other database server information.

Syntax

```
dbsrv17 -c size[ k | m | g | p ] ...
```

Applies to

All operating systems.

Remarks

The amount of memory available for use as a database server cache is one of the key factors controlling performance. You can set the initial amount of cache memory using the -c server option. The more cache memory that can be given to the database server, the better its performance.

The `size` is the amount of memory, in bytes. Use k, m, or g to specify units of kilobytes, megabytes, or gigabytes, respectively.

The unit p is a percentage either of the total physical system memory, or of the maximum supported cache size, whichever is lower. The maximum supported cache size depends on the operating system. For example:

- 2.5 GB for Microsoft Windows 32-bit Advanced Server, Enterprise Server, and Datacenter Server
- 3.5 GB for the 32-bit database server running on Microsoft Windows x64 Edition
- 1.5 GB on all other 32-bit Microsoft Windows operating systems
- On 64-bit database servers, the cache size can be considered unlimited

If you use p, the argument is a percentage. You can use % as an alternative to p, but in Microsoft Windows batch files and some alternate Microsoft Windows command shells, you must escape the % character by doubling it. For example, to set the initial cache size to 50 percent of the physical system memory from a batch file, use a command like the following:

```
dbspawn dbsrv17 -c 50%% ...
```

> **i** Note
>
> On Microsoft Windows 10, the amount of cache memory that you can specify is limited by the total amount of virtual memory available (the sum of physical memory and the pagefile size) minus the amount that has already been committed by other applications. The Microsoft Windows 10 Task Manager shows this memory as *Committed m/n GB* on the *Memory* page of the *Performance* tab.
>
> For example, 6.2/24.4 GB indicates that a total of 6.2 GB of memory is currently committed and that there is another 24.4 - 6.2 or 18.2 GB available that can be committed. This example indicates that -c 20G may exceed the amount of process memory that can be committed by the database server. However, the second number can be increased by Microsoft Windows if the page file can be grown. In any case, system performance can be impaired if you are close to allocating all of usable memory to the cache.
>
> The amount of Committed memory can be very dynamic. It depends on how other processes memory requirements change, so a cache size that worked yesterday may not work today.
>
> If the amount of cache memory requested exceeds the amount of physical memory available, then paging is required and performance is not optimal. The database server logs a message when this situation occurs.

If no -c option is provided, the database server calculates the initial cache allocation as follows:

Microsoft Windows

The formula is as follows:

```
max( 2 MB, min( dbsize, 0.25*TotalPhysicalMemory ) );
```

The `dbsize` is the total size of the database file or files started, and `TotalPhysicalMemory` is the total amount of physical memory on the computer.

UNIX and Linux

At least 8 MB.

```
max( 8 MB, min( 0.1*( physical-memory + available-swap ) , database-size*1.1 )
```

> **Note**
>
> If you attempt to set your initial or minimum cache sizes to a value that is less than one eighth of the maximum cache size, the initial and minimum cache sizes are automatically increased relative to the maximum cache size.

If you disable dynamic cache resizing (-ca option), then the cache size that is used may be restricted by the amount of memory that is available.

The database server messages window displays the size of the cache at startup and you can use the following statement to obtain the current size of the cache:

```
SELECT PROPERTY( 'CurrentCacheSize' );
```

Example

The following example starts a database server named **myserver** that starts with a cache size of 3 MB and loads the sample database:

```
dbsrv17 -c 3m -n myserver "%SQLANYSAMP17%\demo.db"
```

Related Information

Dynamic Cache Sizing [page 1443]
Dynamic Cache Sizing on UNIX/Linux [page 1444]
Tip: Use the Cache to Improve Performance [page 1440]
-ca Database Server Option [page 403]
-cc Database Server Option [page 404]
-ch Database Server Option [page 407]
-chx Database Server Option [page 409]
-cl Database Server Option [page 410]
-cr Database Server Option [page 413]
-cs Database Server Option [page 414]

-cv Database Server Option [page 415]

1.4.1.6 -ca Database Server Option

Enforces a static cache size.

Syntax

```
dbsrv17 -ca 0 ...
```

Applies to

All operating systems.

Remarks

You can disable automatic cache size tuning by specifying `-ca 0` option. If you do not include the `-ca 0` option, the database server automatically increases the cache size. If you specify this option, the cache size is still adjusted if the database server would otherwise run into an error indicating that the dynamic memory is exhausted.

This server option must only be used in the form `-ca 0`.

Example

```
dbsrv17 -c 40P -ca 0 -n myserver "%SQLANYSAMP17%\demo.db"
```

Related Information

Dynamic Cache Sizing [page 1443]
-c Database Server Option [page 400]
-cc Database Server Option [page 404]
-ch Database Server Option [page 407]
-chx Database Server Option [page 409]
-cl Database Server Option [page 410]
-cr Database Server Option [page 413]

-cs Database Server Option [page 414]
-cv Database Server Option [page 415]

1.4.1.7 -cc Database Server Option

Collects information about database pages to be used for cache warming the next time the database is started.

Syntax

```
dbsrv17 -cc{ + | - } ...
```

Default

By default, page collection is turned on.

Applies to

All operating systems.

Allowed Values

-cc+ Turns on page collection.
-cc- Turns off page collection.

Remarks

When collection is turned on, the database server keeps track of each database page that is requested. Collection stops when the maximum number of pages has been collected, the database is shut down, or the collection rate falls below the minimum value. You cannot configure the maximum number of pages collected or specify the value for the collection rate (the value is based on cache size and database size). Once collection stops, information about the requested pages is recorded in the database so those pages can be used to warm the cache the next time the database is started with the -cr option. Collection of referenced pages is turned on by default.

Related Information

Cache Warming [page 1445]
-c Database Server Option [page 400]
-ca Database Server Option [page 403]
-ch Database Server Option [page 407]
-chx Database Server Option [page 409]
-cl Database Server Option [page 410]
-cr Database Server Option [page 413]
-cs Database Server Option [page 414]
-cv Database Server Option [page 415]

1.4.1.8 -cdb Database Server Option

Starts the Cockpit.

Syntax

```
dbsrv17 -cdb option [;...]
option:
DBF={ AUTO | cockpit-database-filename }
START={ YES | NO }
```

On UNIX and Linux, quotation marks are required for the -cdb value when more than one `option` is supplied.

On Windows, quotation marks are required for the -cdb value when space characters appear in`cockpit-database-filename`.

Applies to

All operating systems where the remote data access feature is supported.

Allowed Values

DBF option

Specify the database file where Cockpit configuration settings, such as alert thresholds, and alert histories are saved. If the file does not exist, then it is created. Specify the full path for the file. Otherwise if you use a relative path, it is read relative to the current working directory (or if disk sandboxing is enabled, then the relative path is read relative to the directory where the main database file is located).

This file becomes the default Cockpit database file for the Cockpit for the duration of the database server or until a different Cockpit database file is specified by using the sa_server_option system procedure.

Alternatively, specify AUTO to run the Cockpit using a temporary database. When the Cockpit stops, its temporary database is deleted, so any changes made within the Cockpit are not saved. This value is the default.

START option

Start the Cockpit when the database server starts. This behavior is the default when the DBF option is specified.

Remarks

Start the Cockpit when you start the database server so that it is available whenever you must connect to it. You can also use the sa_server_option system procedure to start, stop, and change settings for the Cockpit on a running server. The Cockpit runs on the same database server that it monitors and uses a database file to store configuration settings.

Use HTTPS to encrypt the communication between the Cockpit and the web browser by starting an HTTPS connection listener (-xs server option). If you do not start an HTTPS connection listener, then the Cockpit starts an unsecured HTTP connection listener. If you use an unsecured HTTP connection listener, then it is easy for a third party to observe communications between the Cockpit and web browser to obtain highly sensitive information, including database user names and passwords that you use to connect to the Cockpit.

When you specify the -xs database server to create an HTTPS connection listener, you can limit the connection listener to the Cockpit by specifying DBN="SQLACockpit".

The Cockpit requires that the database server accept local shared memory connections in addition to HTTP or HTTPS connections. If the database server uses the -ec TLS database server option, then specify the -es database server option.

Use the CockpitURL database server property to obtain the URL to access the Cockpit. For example:

```
SELECT property( 'CockpitURL' );
```

Use the CockpitDB database server property to obtain the current Cockpit settings.

Example

- The following command starts the sample database and the Cockpit. It uses the -xs database server option to create an HTTPS connection listener that can be used by the Cockpit, sample database and any other database that runs on the database server.

  ```
  dbsrv17 -cdb "DBF=C:\temp\cockpitsettings.db" -xs "HTTPS(PORT=443;IDENTITY=C:
  \Users\Public\Documents\SQL Anywhere
      17\Samples\Certificates\rsaserver.id;IDENTITY_PASSWORD=test)" "C:\Users\Public
  \Documents\SQL Anywhere
      17\Samples\demo.db"
  ```

- The following command starts the sample database and the Cockpit. It uses the DBN parameter with the -xs database server option to create an HTTPS connection listener that can only be used by the Cockpit.

  ```
  dbsrv17 -cdb "DBF=C:\temp\cockpitsettings.db" -xs
  "HTTPS(PORT=443;DBN=SQLACockpit;IDENTITY=C:\Users\Public\Documents\SQL Anywhere
      17\Samples\Certificates\rsaserver.id;IDENTITY_PASSWORD=test)" "C:\Users\Public
  \Documents\SQL Anywhere
      17\Samples\demo.db"
  ```

- The following command starts the sample database and the Cockpit. It uses the -xs database server option to create an HTTPS connection listener. This command also encrypts TCPIP communication between the database server and its clients using TLS encryption., while allowing local shared memory connections.

  ```
  dbsrv17 -cdb "DBF=C:\temp\cockpitsettings.db" -xs
  "HTTPS(PORT=443;DBN=SQLACockpit;IDENTITY=C:\Users\Public\Documents\SQL Anywhere
      17\Samples\Certificates\rsaserver.id;IDENTITY_PASSWORD=test)" -ec
  "TLS(IDENTITY=C:\Users\Public\Documents\SQL Anywhere
      17\Samples\Certificates\rsaserver.id;IDENTITY_PASSWORD=test)" -es "C:\Users
  \Public\Documents\SQL Anywhere
      17\Samples\demo.db"
  ```

- The following command sets the default Cockpit database file for the Cockpit, but does not start it.

  ```
  -cdb "START=NO;DBF=cockpitsettings"
  ```

  Later, you can start the Cockpit without specifying the file name and the default Cockpit database is used. For example, `CALL sa_server_option( 'CockpitDB', 'START=YES' );`.

> ⚠ Caution
>
> These examples use a sample identity file that is intended only for testing and development. Do not use it on a production system. Instead, replace it with your own certificate before deploying your application.

Related Information

SQL Anywhere Cockpit [page 1419]
Starting and Connecting to the Cockpit (sa_server_option System Procedure) [page 1423]
-xs Database Server Option [page 529]
sa_server_option System Procedure
List of Database Server Properties [page 900]
Cockpit Security [page 1429]

1.4.1.9 -ch Database Server Option

Sets a maximum cache size, as a limit to automatic cache growth.

> ⁖ Syntax

```
dbsrv17 -ch size[ k | m | g | p ] ...
```

Applies to

All operating systems.

Remarks

This option limits the size of the database server cache during automatic cache growth. By default the upper limit is approximately the lower of the maximum cache size and 90% of the total physical memory of the computer.

The `size` is the amount of memory, in bytes. Use k, m, or g to specify units of kilobytes, megabytes, or gigabytes, respectively.

The unit p is a percentage either of the physical system memory, or of the maximum supported cache size, whichever is lower. The maximum supported cache size depends on the operating system. For example:

- 2.5 GB for Windows 32-bit Advanced Server, Enterprise Server, and Datacenter Server
- 3.5 GB for the 32-bit database server running on Windows x64 Edition
- 1.5 GB on all other 32-bit Windows operating systems
- On 64-bit database servers, the cache size can be considered unlimited

If you use p, the argument is a percentage. You can use % as an alternative to p, but not on Windows operating systems unless you escape the % character. To set the initial cache size to 50 percent of the physical system memory, run the following command:

```
dbsrv17 -ch 50%% ...
```

If you specify a cache size smaller than 64 MB with -ch, the database server adjusts the maximum cache to 64 MB. Specifying a maximum cache size that is smaller than 64 MB is not recommended, but it can be done by using the -chx option.

Example

The following example starts a database server named silver that has a maximum cache size of 64 MB and loads the sample database:

```
dbsrv17 -ch 64m -n silver "%SQLANYSAMP17%\demo.db"
```

Related Information

Dynamic Cache Sizing [page 1443]
-c Database Server Option [page 400]
-ca Database Server Option [page 403]

-cc Database Server Option [page 404]
-chx Database Server Option [page 409]
-cl Database Server Option [page 410]
-cr Database Server Option [page 413]
-cs Database Server Option [page 414]
-cv Database Server Option [page 415]

1.4.1.10 -chx Database Server Option

Sets a maximum cache size as a limit to automatic cache growth.

Syntax

```
dbsrv17 -chx size[ k | m | g | p ] ...
```

Default

Minimum of 512 MB of address space is reserved for non-cache use

Applies to

All operating systems.

Remarks

Caution

When setting a large cache size on 32-bit servers, you should only use the -chx option if you have performed testing to ensure that using additional address space does not affect other components that use server address space, such as:

- DLLs that the server must load
- ODBC drivers for remote table access
- network packet buffers
- external stored procedures
- non-cache memory allocations

If you need a large cache, use the 64-bit version of the database server on a 64-bit operating system.

The 32-bit database server normally limits the cache size so that at least 512 MB of address space is reserved for use outside the cache. To specify a maximum cache size that reserves less address space for non-cache, use the -chx option. Using larger cache sizes may lead to database server instability. This option should be used with caution.

The maximum supported cache size depends on the operating system. For example:

- 2.5 GB for Windows 32-bit Advanced Server, Enterprise Server, and Datacenter Server
- 3.5 GB for the 32-bit database server running on Windows x64 Edition
- 1.5 GB on all other 32-bit Windows operating systems
- On 64-bit database servers, the cache size can be considered unlimited

The `size` is the amount of memory, in bytes. Use k, m, or g to specify units of kilobytes, megabytes, or gigabytes, respectively.

The unit p is a percentage either of the physical system memory, or of the maximum cache size, whichever is lower. You can use % as an alternative to p, but on Windows, which uses % as an environment variable escape character, you must escape the % character.

The -chx option can be used to specify a maximum cache size of less than 64 MB, but using a maximum cache of this size is not recommended.

Related Information

Tip: Use the Cache to Improve Performance [page 1440]
-c Database Server Option [page 400]
-ca Database Server Option [page 403]
-cc Database Server Option [page 404]
-ch Database Server Option [page 407]
-cl Database Server Option [page 410]
-cr Database Server Option [page 413]
-cs Database Server Option [page 414]
-cv Database Server Option [page 415]

1.4.1.11 -cl Database Server Option

Sets a minimum cache size as a lower limit to dynamic cache resizing.

Syntax

```
dbsrv17 -cl size[ k | m | g | p ] ...
```

Default

2 MB on Windows

8 MB on UNIX and Linux

Applies to

All operating systems.

Remarks

This option sets a lower limit to the cache. If -c is specified, and -cl is not specified, then the minimum cache size is set to the initial cache size (the -c setting). If neither -c nor -cl is set, the minimum cache is set to a low, constant value, so that the cache can shrink if necessary. On Windows platforms, this value is 2 MB.

The `size` is the amount of memory, in bytes. Use `k`, `m`, or `g` to specify units of kilobytes, megabytes, or gigabytes, respectively.

The unit p is a percentage either of the physical system memory, or of the maximum supported cache size, whichever is lower. The maximum supported cache size depends on the operating system. For example:

- 2.5 GB for Windows 32-bit Advanced Server, Enterprise Server, and Datacenter Server
- 3.5 GB for the 32-bit database server running on Windows x64 Edition
- 1.5 GB on all other 32-bit systems
- On 64-bit database servers, the cache size can be considered unlimited

If you use p, the argument is a percentage. You can use % as an alternative to p, but on Windows, which uses % as an environment variable escape character, you must escape the % character.

For example, to set the minimum cache size to 50 percent of the physical system memory, run the following command:

```
dbsrv17 -cl 50%% ...
```

> **i Note**
>
> If you attempt to set your initial or minimum cache sizes to a value that is less than one eighth of the maximum cache size, the initial and minimum cache sizes are automatically increased relative to the supported cache size.

Example

The following example starts a database server named silver that has a minimum cache size of 5 MB and loads the database file `example.db`:

```
dbsrv17 -cl 5m -n silver "c:\example.db"
```

Related Information

Dynamic Cache Sizing [page 1443]
-c Database Server Option [page 400]
-ca Database Server Option [page 403]
-cc Database Server Option [page 404]
-ch Database Server Option [page 407]
-chx Database Server Option [page 409]
-cr Database Server Option [page 413]
-cs Database Server Option [page 414]
-cv Database Server Option [page 415]

1.4.1.12 -cp Database Server Option

Specifies a set of directories or JAR files in which to search for classes.

Syntax

```
dbsrv17 -cp location[ ;location ... ] ...
```

On UNIX and Linux, quotation marks are required for the -cp value when more than one `location` is supplied.

On Windows, quotation marks are required for the -cp value when space characters appear in `location`.

Applies to

All operating systems.

Remarks

All classes and JAR files that are being used with Java in the database must be installed in the database. When you store the classes and JAR files within the database, the database can be easily moved to a different

computer or operating system. Another benefit of installing classes and JAR files into the database is that the database server's class loader can fetch the classes and resources from the database, allowing each connection that is using Java in the database to have its own instance of these classes and its own copy of static variables within these classes.

However, when a class or JAR file must be loaded by the system class loader, it can be specified with the java_class_path database option or the -cp database server option. Both options add classes and JAR files to the classpath that the database server builds for launching the Java VM. The database server does not use the CLASSPATH as defined by the environment variable of that name. The java_class_path database option is useful when the server is running multiple databases and each database has a different set of JARs and directories that need to be loaded by the system class loader. The -cp database server option is useful when all databases on the server require the same classes or JAR files.

The JAR files required to support the database server as an OData server are always also included in the server's class path.

Related Information

Java in the Database
java_class_path Option [page 748]

1.4.1.13 -cr Database Server Option

Reloads (warms) the cache with database pages using information collected the last time the database was run.

Syntax

```
dbsrv17 -cr{ + | - } ...
```

Default

By default, cache warming is turned on.

Applies to

All operating systems.

Allowed Values

-cr+ Turns on cache warming.
-cr Turns on cache warming.
-cr- Turns off cache warming.

Remarks

You can instruct the database server to warm the cache using pages that were referenced the last time the database was started (page collection is turned on using the -cc option). Cache warming is turned on by default. When a database is started, the server checks the database to see if it contains a collection of pages requested the last time the database was started. If the database contains this information, the previously referenced pages are then loaded into the cache.

Warming the cache with pages that were referenced the last time the database was started can improve performance when the same query or similar queries are executed against a database each time it is started.

Related Information

Cache Warming [page 1445]
-cc Database Server Option [page 404]
-cl Database Server Option [page 410]
-cs Database Server Option [page 414]
-cv Database Server Option [page 415]

1.4.1.14 -cs Database Server Option

Displays statistics related to dynamic cache sizing in the database server messages window.

Syntax

```
dbsrv17 -cs ...
```

Applies to

All operating systems.

Remarks

For troubleshooting purposes, this option displays statistics in the database server messages window that database server is using to determine how to tune size of the cache.

Related Information

Cache Warming [page 1445]
-c Database Server Option [page 400]
-ca Database Server Option [page 403]
-cc Database Server Option [page 404]
-ch Database Server Option [page 407]
-chx Database Server Option [page 409]
-cl Database Server Option [page 410]
-cr Database Server Option [page 413]
-cv Database Server Option [page 415]
sa_server_option System Procedure

1.4.1.15 -cv Database Server Option

Controls the appearance of messages about cache warming in the database server messages window.

Syntax

```
dbsrv17 -cv{ + | - } ...
```

Default

Cache warming messages are suppressed.

Applies to

All operating systems.

Allowed Values

-cv+ Cache warming messages are displayed.
-cv Cache warming messages are displayed.
-cv- Cache warming messages are suppressed.

Remarks

When -cv+ is specified, a message appears in the database server messages window when any of the following cache warming activities occur:

- collection of requested pages starts or stops (controlled by the -cc server option)
- page reloading starts or stops (controlled by the -cr server option)

Example

The following command starts the database `mydatabase.db` with database page collection and page loading turned on, and logs messages about these activities to the database server messages window:

```
dbsrv17 -cc+ -cr+ -cv+ mydatabase.db
```

Related Information

Cache Warming [page 1445]
-c Database Server Option [page 400]
-ca Database Server Option [page 403]
-cc Database Server Option [page 404]
-ch Database Server Option [page 407]
-cl Database Server Option [page 410]
-cr Database Server Option [page 413]
-cs Database Server Option [page 414]

1.4.1.16 -dt Database Server Option

Specifies the directory where temporary files are stored.

Syntax

```
dbsrv17 -dt temp-file-dir ...
```

Applies to

All operating systems, except shared memory connections on UNIX and Linux.

Remarks

The database server creates two types of temporary files:

Database server-related temporary files (created on all platforms)

You can use the -dt option to specify a directory for database server-related temporary files. If you do not specify this option when starting the database server, the database server examines the following environment variables, in the order shown, to determine the directory in which to place the temporary file.

Windows/UNIX/Linux
1. SATMP 2. TMP 3. TMPDIR 4. TEMP

If none of the environment variables are defined, the database server places its temporary file in the current directory on Windows, and in the `/tmp` directory on UNIX and Linux.

Communications-related temporary files (created only on UNIX and Linux for both the client and the database server)

Temporary files for communications on UNIX and Linux are not placed in the directory specified by -dt. Instead, the database server examines the following environment variables, in the order shown, to determine the directory in which to place the temporary file:

UNIX/Linux
1. SATMP 2. TMP 3. TMPDIR 4. TEMP

If none of the environment variables are defined, the database server places its temporary file in the `/tmp` directory on UNIX and Linux.

On UNIX and Linux, both the client and the database server must set SATMP to the same value when connecting via shared memory.

Example

To locate the database server-related temporary files, use the DB_PROPERTY system function with the TempFileName property:

```
SELECT DB_PROPERTY ( 'TempFileName' );
```

Related Information

Database File Types [page 280]
SATMP Environment Variable [page 578]
sa_disk_free_space System Procedure
temp_space_limit_check Option [page 845]
Troubleshooting: Database Server Startup [page 1023]

1.4.1.17 -ec Database Server Option

Uses transport layer security (TLS) or simple obfuscation to encode communication protocol packets (such as DBLIB and ODBC) transmitted to and from all clients.

Syntax

```
dbsrv17 -ec encoding-option [,...]
```

```
Specify at least one of the supported options in a comma-separated list.
encoding-option :
{ NONE
  | SIMPLE
  | TLS (
    [ FIPS={ ON | OFF } ; ]
    IDENTITY=server-identity-filename;
    IDENTITY_PASSWORD=password
    MIN_TLS_VERSION=ver
    )
```

On UNIX and Linux, quotation marks are required when specifying the TLS options for the -ec option:

```
dbsrv17 -ec "TLS(encoding-option [;...])"
```

Allowed Values

NONE

The database server accepts connections that aren't encrypted. These connections can be remote connections, or they can be local connections over shared memory.

SIMPLE

The database server accepts connections that are encoded using simple obfuscation. The database server also accepts connections that are not encoded and upgrades them to simple obfuscation.

When both the SIMPLE and NONE parameters are specified, the database server accepts connections that are not encoded, but does not upgrade them.

TLS

The database server accepts connections that are encrypted using the Transport Layer Security RSA algorithm. The TLS parameter accepts the following arguments:

FIPS

For FIPS-certified TLS, specify FIPS=ON. FIPS-certified TLS uses a separate library, but is compatible with uncertified TLS.

FIPS-certified RSA encryption requires a separate license.

IDENTITY=server-identity-filename

The path and file name of the server identity certificate. You must generate your certificates using the RSA algorithm.

IDENTITY_PASSWORD=password

The password for the private key contained in the identity file. This password was defined when the server identity certificate was created.

MIN_TLS_VERSION=ver

The minimum downgrade TLS version. The supported values for `ver` are *1.0* (the default), *1.1*, and *1.2*. The period is optional, so you can also specify *10*, *11*, or *12*.

Default

The default is NONE, SIMPLE.

When both the SIMPLE and NONE parameters are specified, the database server accepts connections that are encoded using simple obfuscation as well as connections that are not encoded.

Applies to

Applies to all operating systems.

Remarks

Use this option to secure communication packets between client applications and the database server. By default, communication packets aren't encrypted, which poses a potential security risk. If you are concerned about the security of network packets, then use the -ec option.

The -ec option instructs the database server to accept only connections that are encrypted using one of the specified types. Specify at least one of the supported parameters in a comma-separated list.

The -ec NONE option instructs the database server to accept both remote and local shared memory connections that are not encrypted.

The -es SIMPLE option uses simple obfuscation that doesn't provide server verification, strong encryption, or other features of transport layer security.

If you specify -ec TLS or -ec SIMPLE and you want to accept shared memory connections (which are not encoded/encrypted), then you must also specify the -es database server option.

All external environments connect back to the database server over shared memory. If the database server is configured to accept only encrypted connections, then the external environment will fail to connect. In this case, make sure to include the -es option with the other database server start options to enable shared memory connections.

The client's and the database server's encryption settings must match or the connection fails except in the following cases:

- If -ec SIMPLE is specified on the database server, but -ec NONE is not, then connections that do not specify the Encryption connection parameter and connections that specify Encryption=NONE can connect and are automatically upgraded to use simple obfuscation.
- If the database server specifies uncertified TLS encryption and the client specifies FIPS-certified TLS encryption, or vice versa, then the connection succeeds. In these cases, the Encryption connection property returns the value specified by the database server.

For TDS communication protocol connections, such as Java applications using jConnect, only the TLS option is supported. Simple obfuscation is not supported for TDS connections. To prevent unencrypted TDS connections, you must set the TDS protocol option to NO or ENCRYPTED.

Encryption affects performance only marginally.

The `dbrsa17.dll` (`libdbrsa17_r.so` on UNIX and Linux) file contains the TLS code used for encryption and decryption. The `dbfips17.dll` (`libdbfips17_r.so` on Linux) file contains the FIPS-certified version of the TLS algorithm. When you connect to the database server, if the appropriate file cannot be found, or if an error occurs, then a message appears in the database server messages window. The database server doesn't start if the specified types of encryption cannot be initiated.

Example

The following example specifies that connections with no encryption and simple obfuscation are allowed.

On Windows:

```
dbsrv17 -ec NONE,SIMPLE -x tcpip mydemo.db
```

On UNIX and Linux:

```
dbsrv17 -ec "NONE,SIMPLE" -x tcpip mydemo.db
```

The following example starts a database server that uses TLS encryption and the identity certificate `rsaserver.id`.

On Windows:

```
dbsrv17 -ec TLS(IDENTITY=rsaserver.id;IDENTITY_PASSWORD=test) -x tcpip mydemo.db
```

On UNIX and Linux:

```
dbsrv17 -ec "TLS(IDENTITY=rsaserver.id;IDENTITY_PASSWORD=test)" -x tcpip
mydemo.db
```

The following example starts a database server using FIPS-certified TLS encryption algorithms and the identity certificate `rsaserver.id`. If `rsaserver.id` is expired, then it is still accepted by the database server.

```
dbsrv17 -ec TLS(FIPS=ON;IDENTITY=rsaserver.id;IDENTITY_PASSWORD=test; -x tcpip
mydemo.db
```

The following example connects to this server and displays the text `rsa_tls_fips` at the command prompt.

```
dbisql -c
"UID=DBA;PWD=passwd;ENC=TLS(TRUSTED_CERTIFICATE=rsaroot.crt;SKIP_CERTIFICATE_NAME
_CHECK=ON);Server=mydemo;LINKS=TCPIP" select connection_property('encryption')
```

> **i Note**
>
> The sample certificates referenced here are installed in the `samples\certificates` folder.

Related Information

Transport Layer Security [page 1727]
Database Server with Transport Layer Security [page 1746]
Digital Certificates [page 1737]
TDS Protocol Option (Server Side Only) [page 241]
Encryption (ENC) Connection Parameter [page 80]
-ek Database Option [page 551]
-ep Database Server Option [page 423]
-es Database Server Option [page 424]
DatabaseKey (DBKEY) Connection Parameter [page 70]
FIPS Protocol Option [page 205]
-fips Database Server Option [page 426]

1.4.1.18 -edi Database Server Option

Turns on database isolation.

Syntax

```
dbsrv17 -edi ...
```

Default

By default, enable database isolation is turned off.

Applies to

Applies to all operating systems.

Remarks

When this option is specified all databases behave as though they are the only database running on the database server. Connections to one database cannot query properties or connection information for any other database running on the same database server. Connections cannot stop or start other databases running on the same database server.

Note

Use the database_isolation secure feature to temporarily turn off database isolation for the current connection.

When database isolation is turned on, applications can still connect to the utility database with the appropriate utility user ID and password and can perform utility actions. However, starting and stopping databases while connected to the utility database is restricted: applications must use the database_isolation secure feature to start and stop databases.

Related Information

-sf Database Server Option [page 493]

1.4.1.19 -ep Database Server Option

Prompts the user for the encryption key upon starting a strongly encrypted database.

Syntax

```
dbsrv17 -ep ...
```

Applies to

All operating systems.

Remarks

The -ep option instructs the database server to display a window where the user enters the encryption key for databases started on the command line that require an encryption key. This server option provides an extra measure of security by never allowing the encryption key to be seen in clear text.

When used with the database server, the user is prompted for the encryption key when the following are all true:

- The -ep option is specified
- An encryption key is required to start a database
- The database server isn't a Windows service or a daemon (UNIX and Linux)

To secure communication packets between client applications and the database server, use the -ec server option and transport layer security.

Example

The user is prompted for the encryption key when the `myencrypted.db` database is started:

```
dbsrv17 -ep -x tcpip myencrypted.db
```

Related Information

Transport Layer Security [page 1727]
Database Server with Transport Layer Security [page 1746]
-ec Database Server Option [page 418]
-ek Database Option [page 551]

Encryption (ENC) Connection Parameter [page 80]
DatabaseKey (DBKEY) Connection Parameter [page 70]

1.4.1.20 -es Database Server Option

Allows unencrypted connections over shared memory.

Syntax

```
dbsrv17 -ec encryption-options -es ...
```

Applies to

All operating systems.

Remarks

This option is only effective when specified with the -ec option. The -es option instructs the database server to accept local, unencrypted, command-sequence connections over shared memory. In contrast the -ec NONE option instructs the database server to accept both remote and local, command-sequence connections that are not encrypted.

Connections over TCP/IP must use an encryption type specified by the -ec option. The -es option is useful in situations where you want remote clients to use encrypted connections, but for performance reasons you also want to access the database from the local computer with an unencrypted connection.

Example

- For example, the SQL Anywhere Cockpit requires shared memory connections. To run the Cockpit on a database server that is encrypted with TLS, start the database server using `-ec TLS(...) -es`.
- The following example specifies that connections with simple obfuscation and unencrypted connections over shared memory are allowed.

  ```
  dbsrv17 -ec SIMPLE -es -x tcpip c:\mydemo.db
  ```

Related Information

Database Server with Transport Layer Security [page 1746]

-ec Database Server Option [page 418]

1.4.1.21 -fc Database Server Option

Specifies the file name of a DLL (or shared object on Unix) containing the File System Full callback function.

Syntax

```
dbsrv17 -fc filename ...
```

Applies to

All operating systems.

Remarks

This option can be used to notify users, and possibly take corrective action, when a file system full condition is encountered. If you use the -fc option, the database server attempts to load the specified DLL and resolve the entry point of the callback function during startup. If the database server cannot find both the DLL and the entry point, the database server returns an error and shuts down. The DLL is user-supplied and can use the callback to, among other things, invoke a batch file (or shell script on Unix) you have supplied to take diagnostic or corrective action. Alternatively, the callback function itself can perform such an action.

A sample disk full callback function is located in *%SQLANYSAMP17%*`\SQLAnywhere\DiskFull`.

The database server searches for the callback function DLL in the same locations as it searches for other DLLs and files.

When the database server detects a disk full condition, it invokes the callback function (if one has been provided), passing it the following information:

- the name of the dbspace where the condition was triggered
- the operating system-specific error code from the failed operation

The return code from the call to xp_out_of_disk indicates whether the operation that caused the condition should be aborted or retried. If a non-zero value is returned, the operation is aborted, otherwise it is retried. The callback function is invoked repeatedly as long as it returns zero and the file system operation fails.

On Microsoft Windows platforms, if the database server is started with a database server messages window (neither -qi nor -qw have been specified), and a callback DLL is not provided, a window appears when a disk full condition occurs. This window contains the dbspace name and error code, and allows the user to choose whether the operation that caused the disk full condition should be retried or aborted.

On all other operating systems, when -fc isn't specified and a disk full condition is encountered, a fatal error occurs.

You can create system events to track the available disk space of devices holding the database file, the transaction log file, or the temporary file and alert administrators of a disk space shortage.

Example

When the database server starts, it attempts to load the `diskfull.dll` DLL.

```
dbsrv17 -fc diskfull.dll -n my_server
```

Related Information

How SQL Anywhere Locates Files [page 587]
Callback Functions
System Events [page 1001]
CREATE EVENT Statement
max_temp_space Option [page 770]
temp_space_limit_check Option [page 845]

1.4.1.22 -fips Database Server Option

Sets the database server to use FIPS-certified encryption for strong database and communication encryption.

Syntax

```
dbsrv17 -fips ...
```

Applies to

Windows and Linux.

Remarks

This option applies to strong database encryption, client/server transport layer security, and web services transport layer security.

FIPS-certified encryption requires a separate license.

For strong database encryption, the -fips option causes new databases to use the FIPS-certified equivalent of AES and AES256 if they are specified in the ALGORITHM clause of the CREATE DATABASE statement.

When the database server is started with -fips, you can run databases encrypted with AES, AES256, AES_FIPS, or AES256_FIPS encryption, but not databases encoded with simple obfuscation. Unencrypted databases can also be started on the server when -fips is specified.

The SQL Anywhere FIPS-certified AES/RSA encryption components must be installed on any computer used to run a database encrypted with AES_FIPS or AES256_FIPS.

For transport layer security, the -fips option causes the server to use the FIPS-certified encryption algorithm, even if RSA is specified.

For transport layer security for web services, the -fips option causes the server to use FIPS-certified HTTPS, even if HTTPS is specified instead of HTTPS_FIPS.

When you specify -fips, the ENCRYPT and HASH functions use the FIPS-certified encryption algorithm. For example, if you specify SHA1, then SHA1_FIPS is used instead.

When you specify -fips, password hashing uses the SHA256_FIPS algorithm instead of the SHA256 algorithm.

Password hashing refers to the way passwords are stored in the database. It does not refer to how passwords are exchanged/encrypted between client and server. In this case, the AES256/AES256_FIPS encryption algorithm is used.

When you specify -fips and then you start an email session using the xp_startsmtp system procedure, then the database server always uses PLAIN authentication to authenticate the IDs and passwords with the SMTP server. The -fips database server option does not support CRAM_MD5 authentication.

Related Information

Simple Obfuscation Versus Strong Encryption [page 1698]
Separately Licensed Components
Transport Layer Security [page 1727]
SQL Anywhere Web Services Encryption [page 1751]
-ec Database Server Option [page 418]
ENCRYPT Function [String]
HASH Function [String]

1.4.1.23 -ga Database Server Option

Unloads the database after the last non-HTTP client connection disconnects.

Syntax

```
dbsrv17 -ga ...
```

Applies to

All operating systems.

Remarks

Specifying this option on the network server causes each database to be unloaded after the last non-HTTP client connection disconnects. In addition to unloading each database after the last non-HTTP connection disconnects, the database server shuts down when the last database is stopped.

If the only connection to a database is an HTTP connection, and the database is configured to stop automatically, when the HTTP connection disconnects, the database is not unloaded. As well, if you specify the -ga option, and the database has an HTTP connection and a command sequence or TDS connection, when the last command sequence or TDS connection disconnects, the database stops automatically, and any HTTP connections are dropped.

Related Information

Database Rebuilds
AutoStop (ASTOP) Connection Parameter [page 54]

1.4.1.24 -gb Database Server Option

Sets the server process priority class.

Syntax

(Windows only)

```
dbsrv17 -gb { IDLE | NORMAL | HIGH | MAXIMUM } ...
```

(UNIX/Linux only)

```
dbsrv17 -gb level ...
```

Allowed Values

Windows

On Windows, NORMAL and HIGH are the commonly used settings. The value IDLE is provided for completeness. The value MAXIMUM may interfere with the running of your computer.

UNIX/Linux

On UNIX and Linux, `level` is an integer from -20 to 19. The default value on UNIX and Linux is the same as the nice value of the parent process. Lower `level` values represent a more favorable scheduling priority. All restrictions placed on setting a nice value apply to the -gb option. For example, on most UNIX and Linux platforms, only the root user can lower the priority level of a process (for example, changing it from 0 to -1).

Applies to

All operating systems.

1.4.1.25 -gc Database Server Option

Sets the maximum interval between checkpoints.

Syntax

```
dbsrv17 -gc minutes ...
```

Default

60 minutes

Allowed Values

minutes

The default value is the setting of the checkpoint_time database option, which defaults to 60 minutes. If a value of 0 is entered, the default value of 60 minutes is used.

Applies to

All operating systems.

Remarks

Use this option to set the maximum length of time, in minutes, that the database server runs without doing a checkpoint on each database.

Checkpoints generally occur more frequently than the specified time.

Related Information

Checkpoint Logs [page 307]
How the Database Server Decides When to Checkpoint [page 306]
checkpoint_time Option [page 700]

1.4.1.26 -gd Database Server Option

Sets the privileges required to start or stop a database on a running database server.

Syntax

```
dbsrv17 -gd { DBA | ALL | NONE } ...
```

Allowed Values

DBA

Only users with the SERVER OPERATOR system privilege can start or stop databases.

ALL

All users can start or stop databases. Not recommended for network servers that can be accessed by remote clients.

NONE

Starting and stopping databases isn't allowed except when the database server itself is started and stopped.

Default

The default setting is *DBA* for the network database server. Both uppercase and lowercase syntax are allowed.

The default setting is *ALL* for the personal database server.

Applies to

All operating systems.

Remarks

This option specifies the level of privilege required for a user to cause a new database file to be loaded by the database server, or to stop a database on a running database server.

When the option is set to DBA, the client application must use an existing connection to another database running on the same server to start and stop databases. You cannot start a database that is not already running by using the DatabaseFile (DBF) connection parameter.

You can obtain the setting of the -gd option using the StartDBPermission server property:

```
SELECT PROPERTY ( 'StartDBPermission' );
```

The privileges for stopping a database server are specified by the -gk option.

Example

The following steps illustrate how to use the -gd option for the network database server.

1. Start the network database server:

   ```
   dbsrv17 -su passwd -gd DBA -n my_server
   ```
2. Connect to the utility database from Interactive SQL:

   ```
   dbisql -c "UID=DBA;PWD=passwd;DBN=utility_db"
   ```
3. Start a database:

   ```
   START DATABASE 'demo.db';
   ```
4. Connect to the database:

   ```
   CONNECT USING 'DBN=demo;UID=DBA;PWD=sql';
   ```

Related Information

Inheritance of Roles and Privileges [page 1617]
-gk Database Server Option [page 433]
STOP DATABASE Statement

1.4.1.27 -ge Database Server Option

Sets the stack size for external functions.

Syntax

```
dbsrv17 -ge integer ...
```

Default

32 KB

Applies to

Windows.

Remarks

Sets the stack size for threads running external functions, in bytes.

Related Information

Threading Behavior [page 347]

1.4.1.28 -gf Database Server Option

Disables firing of triggers by the server.

Syntax

```
dbsrv17 -gf ...
```

Applies to

All operating systems.

Remarks

The -gf server option instructs the server to disable the firing of triggers, including referential integrity triggers (such as cascading updates and deletes).

Related Information

Triggers
fire_triggers Option [page 736]

1.4.1.29 -gk Database Server Option

Sets the privileges required to stop the database server.

Syntax

```
dbsrv17 -gk { DBA | ALL | NONE } ...
```

Allowed Values

DBA

Only users with the SERVER OPERATOR system privilege can stop the database server.

This value is the default for the network server.

ALL

No privileges are required to shut down the database server.

This value is the default for the personal server.

NONE

The database server cannot be stopped.

Applies to

All operating systems.

Remarks

The -gd database server option applies to the dbstop utility as well as to the following statements:

- ALTER DATABASE `dbname` FORCE START statement.
- STOP DATABASE statement

Related Information

Stop Server Utility (dbstop) [page 1219]

1.4.1.30 -gl Database Server Option

Sets the privilege required to load data using the LOAD TABLE statement, and to unload data using the UNLOAD or UNLOAD TABLE statements.

Syntax

```
dbsrv17 -gl { DBA | ALL | NONE } ...
```

Allowed Values

DBA

For unloading data, you must have the SELECT ANY TABLE system privilege.

For loading data, you must have the LOAD ANY TABLE, ALTER ANY TABLE, or ALTER ANY OBJECT system privilege.

ALL

For unloading data, you must be the owner of the tables, have the SELECT privilege on the tables, or have the SELECT ANY TABLE system privilege.

For loading data, one of the following must be true:

- You are the owner of the table.
- You have LOAD privilege on the table.

- You have the LOAD ANY TABLE system privilege.
- You have the ALTER ANY TABLE system privilege.
- You have the ALTER ANY OBJECT system privilege.

NONE

Data cannot be unloaded or loaded.

Default

DBA is the default value for the network server.

On Windows, the default setting is *ALL* for personal database servers, because the personal database server is running on the current computer, so the user already has access to the file system.

On Unix, the default setting is *DBA* for the personal server and the network database server.

Applies to

All operating systems.

Remarks

Using the UNLOAD TABLE or UNLOAD statement places data in files on the database server computer, and the LOAD TABLE statement reads files from the database server computer.

To control access to the file system using these statements, the -gl server option allows you to control the privileges required to use these statements.

Both uppercase and lowercase syntax are acceptable.

Related Information

LOAD TABLE Statement
UNLOAD Statement

1.4.1.31 -gm Database Server Option

Limits the number of concurrent connections to the database server.

Syntax

```
dbsrv17 -gm integer ...
```

Default

The default value and the connection limit for the personal database server (dbeng17) is 10 connections, of which three connections can only accept standard connections.

The default value for the network database server depends on your license agreement.

Applies to

All operating systems.

Remarks

Defines the connection limit for the database server.

If this number is greater than the number that is allowed under licensing and memory constraints, it has no effect.

The database server allows one extra connection above the connection limit to allow a user with the DROP CONNECTION or SERVER OPERATOR system privilege to connect to the database server and drop other connections.

The database server queues HTTP/HTTPS connection attempts when the database server reaches its HTTP/HTTPS connection limit. While there are connections in the HTTP/HTTPS queue, there is no opportunity for a standard connection to replace an HTTP/HTTPS connection. A user wanting to make a standard connection could wait indefinitely unless some of the database connections are reserved for standard connections. Reserve some of the database's connections for standard connections by specifying the reserved_connections database option.

You can also limit the maximum number of HTTP/HTTPS connections that can be made to a database server by specifying the MaxConnections protocol option.

Related Information

Connection Limits for Databases and Database Servers [page 248]
Limiting Database Connections [page 251]
max_connections Option [page 761]
MaxConnections (MAXCONN) Protocol Option [page 225]

1.4.1.32 -gn Database Server Option

Sets the multiprogramming level of the database server.

Syntax

```
dbsrv17 -gn integer ...
```

Default

20 active tasks

Applies to

All operating systems.

Remarks

The -gn server option sets the initial database server multiprogramming level.

By default, the network database server automatically adjusts the server multiprogramming level in response to changes in the server's workload. You can manually override the setting of the multiprogramming after a network database server has started by using the sa_server_option system procedure and the CurrentMultiprogrammingLevel parameter.

Note

Setting a large `integer` value can reduce the maximum cache size available on 32-bit platforms.

Related Information

Threading [page 344]
Database Server Configuration of the Multiprogramming Level [page 348]
sa_server_option System Procedure
-gna Database Server Option [page 438]
-gnh Database Server Option [page 439]
-gnl Database Server Option [page 440]
sa_server_option System Procedure

1.4.1.33 -gna Database Server Option

Controls automatic tuning of the database server multiprogramming level.

Syntax

```
dbsrv17  -gna { 0 | 1 }
```

Default

1 (Enabled)

Applies to

All operating systems.

Network database server (dbsrv17) only.

Remarks

This option enables and disables dynamic tuning of the database server's multiprogramming level. You can manually change the multiprogramming level while the database server is running using the sa_server_option system procedure.

By default, automatic tuning of the database server's multiprogramming level is enabled.

Related Information

Threading [page 344]
Database Server Configuration of the Multiprogramming Level [page 348]
max_query_tasks Option [page 767]
-gn Database Server Option [page 437]
-gnh Database Server Option [page 439]
-gnl Database Server Option [page 440]
sa_server_option System Procedure

1.4.1.34 -gnh Database Server Option

Sets the maximum multiprogramming level - the maximum number of tasks that the network database server can execute concurrently.

Syntax

```
dbsrv17 -gnh integer ...
```

Default

Four times the -gn database server option value

Applies to

All operating systems.

Network database server (dbsrv17) only.

Remarks

This option sets the maximum multiprogramming level of the network database server. It limits the maximum number of tasks (both user and system requests) that the network database server can execute concurrently. If the network database server receives an additional request while at this limit, the new request must wait until an executing task completes. The number of active tasks that can execute simultaneously depends on the number of network database server threads and the number of logical processors in use.

The maximum number of combined unscheduled and active requests is limited by the -gm network database server option, which limits the number of connections to the network database server.

Setting the -gnh value too high can result in errors because a larger portion of the system's address space is consumed for stack space.

The network database server's kernel uses tasks as the scheduling unit. The execution of any user request requires at least one task. However, a request may cause the scheduling of additional tasks on its behalf. One example of this behavior is if the request involves the execution of an external procedure or function (such as Java or Perl) that in turn makes database requests back into the network database server.

When intra-query parallelism is involved, each access plan component executed in parallel is a task. These tasks count toward the -gnh limit as though they were separate requests. However, tasks created for intra-query parallelism are not reflected in the database properties that track the number of active and inactive requests.

⚠ Caution

A stack of the size specified by -gss is allocated for each database server worker.

The maximum number of workers is specified by the -gnh option for network servers.

For personal servers, the maximum number of workers is specified by the -gn option.

If you set a large -gss value, and a large value for the maximum number of workers, then the database server may not be able to start, or the size of the cache can be limited significantly. For example if you specified -gss 16M and -gnh 100 when starting the network database server, then 1.6 GB of server address space would be reserved for stacks.

Related Information

Threading [page 344]
Database Server Configuration of the Multiprogramming Level [page 348]
max_query_tasks Option [page 767]
-gn Database Server Option [page 437]
-gna Database Server Option [page 438]
-gnl Database Server Option [page 440]
-gss Database Server Option [page 445]
sa_server_option System Procedure

1.4.1.35 -gnl Database Server Option

Sets the minimum database server multiprogramming level.

Syntax

```
dbsrv17  -gnl integer ...
```

Default

The minimum of the value of the -gtc server option and the number of logical CPUs on the computer.

Applies to

All operating systems.

Network database server (dbsrv17) only.

Remarks

Setting the -gnl value ensures that the current multiprogramming level cannot be set lower than this value. In workloads that are intensive followed by a period of idle time, it may be beneficial to increase the -gnl value to improve database server performance when intensive processing re-occurs.

Related Information

Threading [page 344]
Database Server Configuration of the Multiprogramming Level [page 348]
-gtc Database Server Option [page 449]
max_query_tasks Option [page 767]
-gn Database Server Option [page 437]
-gna Database Server Option [page 438]
-gnh Database Server Option [page 439]
sa_server_option System Procedure

1.4.1.36 -gns Database Server Option

Displays statistics for automatic multiprogramming level adjustments in the database server message log.

Syntax

```
dbsrv17  -gns ...
```

Default

Statistics are not displayed

Applies to

All operating systems.

Network database server (dbsrv17) only.

Remarks

Multiprogramming level statistics are recorded in the database server message log only when the -gns option is specified.

You can change the setting for the reporting of multiprogramming level statistics once the database server has started using the sa_server_option system procedure.

To determine whether multiprogramming level statistics are currently being reported, execute the following query:

```
SELECT PROPERTY ( 'AutoMultiProgrammingLevelStatistics' );
```

Multiprogramming level statistics are recorded on the database server message log similar to the following example:

```
MPL[curr=20 prev=20] concurrency[4] throughput[curr=2698 prev=3270]
```

The above example is interpreted as follows:

Statistic	Description
curr=20	Number of workers in the current time interval.
prev=20	Number of workers in the previous time interval.
concurrency[4]	Average number of concurrent (parallel) requests in the current time interval.
throughput[curr=2698]	Number of requests processed in the current time interval.
throughput[prev=3270]	Number of requests processed in the previous time interval.

Related Information

Database Server Configuration of the Multiprogramming Level [page 348]
Database Server Logging [page 336]

sa_server_option System Procedure
-gn Database Server Option [page 437]
-gna Database Server Option [page 438]
-gnh Database Server Option [page 439]
sa_server_option System Procedure
-gnl Database Server Option [page 440]

1.4.1.37 -gp Database Server Option

Sets the maximum allowed database page size.

Syntax

```
dbsrv17 -gp { 2048 | 4096 | 8192 | 16384 | 32768 } ...
```

Default

4096 (if a database server is started with no databases loaded).

Applies to

All operating systems.

Remarks

Database files with a page size larger than the page size of the server cannot be loaded. This option explicitly sets the page size of the server, in bytes.

By default, the server page size is the same as the largest page size of the databases on the command line.

The 2048 page size is deprecated.

On all platforms, if you do not use this option and start a server with no databases loaded, the default value is 4096.

Related Information

Maximum Page Size Considerations [page 333]

1.4.1.38 -gr Database Server Option

Sets the maximum length of time (in minutes) for recovery from system failure.

Syntax

```
dbsrv17 -gr minutes ...
```

Default

Setting of the recovery_time database option, which defaults to 2 minutes.

Applies to

All operating systems.

Remarks

When a database server is running with multiple databases, the recovery time that is specified by the first database started is used unless overridden by this option.

The value specified by the -gr option instructs the database server how often to perform a checkpoint. For example, if you set -gr to 5, then the database server tries to perform checkpoints often enough so that recovery takes no longer than 5 minutes. However, if recovery is necessary, it runs to completion, even if it takes longer than the length of time specified by -gr.

The recovery time includes both the estimated recovery time and the estimated checkpoint time for the database.

Related Information

How the Database Server Decides When to Checkpoint [page 306]
recovery_time Option [page 806]

1.4.1.39 -gss Database Server Option

Sets the stack size per worker in the database server.

Syntax

```
dbsrv17 -gss integer[ k | m ] ...
```

Applies to

All operating systems.

Remarks

The -gss option allows you to lower the memory usage of the database server in environments with limited memory.

The `size` is the amount of memory to use, in bytes. Use k or m to specify units of kilobytes or megabytes, respectively.

⚠ Caution

A stack of the size specified by -gss is allocated for each database server worker. The maximum number of workers is specified by the -gnh option for network servers.

For personal servers, the maximum number of works is specified by the -gn option.

If you set a large -gss value, and a large value for the maximum number of workers, then the database server may not be able to start, or the size of the cache can be limited significantly. For example if you specified -gss 16M and -gnh 100 when starting the network database server, then 1.6 GB of server address space would be reserved for stacks.

The minimum, maximum, and default stack sizes per worker are as follows:

Platform	Minimum	Default	Maximum
Unix (32-bit)	512 KB	512 KB	4 MB
Unix (64-bit)	1 MB	1 MB	8 MB
Windows (32-bit)	64 KB	1 MB	16 MB
Windows (64-bit)	64 KB	4 MB	256 MB

Related Information

Threading [page 344]
Workers on Microsoft Windows and Linux [page 346]

1.4.1.40 -gt Database Server Option

Sets the maximum number of physical processors that can be used (up to the licensed maximum). This option is only useful on multiprocessor systems.

Syntax

```
dbsrv17 -gt num-processors ...
```

Allowed Values

num-processors

This integer can be a value between 1 and the minimum of:

- the number of potential physical processors on the computer, and
- the maximum number of CPUs that the server is licensed for if CPU-licensing is in effect.

If the -gt value specified lies outside this range, the minimum or maximum integer value is imposed.

For the personal database server (dbeng17), the server uses a -gt value of 1.

Applies to

All operating systems. The setting is ignored for all operating systems except Windows, Linux, and Solaris.

Remarks

With per-seat licensing, the network database server uses all CPUs available on the computer. With CPU-based licensing, the network database server uses only the number of processors you are licensed for. The number of CPUs that the network database server can use may also be restricted by your SQL Anywhere edition.

The personal database server is always limited to a single processor.

Physical processors are sometimes referred to as packages or dies, and are the CPUs of the computer. Additional logical processors exist when the physical processors support hyperthreading or are themselves

configured as multiprocessors (often referred to as multi-core processors). The operating system schedules threads on logical processors.

When you specify a value for the -gt option, the database server adjusts its affinity mask (if supported on that hardware platform) to restrict the database server to run on only that number of physical processors. If the database server is licensed for `n` processors, the server, by default, runs on all logical processors (hyperthreads and cores) of `n` physical processors. This behavior can be further restricted with the -gtc option.

The -gt option cannot be used with the -gta option.

Related Information

Editions and Licensing
Threading [page 344]
-gnh Database Server Option [page 439]
-gtc Database Server Option [page 449]
-gta Database Server Option [page 447]
sa_server_option System Procedure

1.4.1.41 -gta Database Server Option

Sets which logical processors the database server can use.

Syntax

```
dbsrv17 -gta logical-processors-to-use,...
```

On UNIX and Linux, quotation marks are required when more than one parameter is supplied:

Allowed Values

logical-processors-to-use

Accepts a comma-delimited list of processor numbers and/or ranges. A range is indicated wth a hyphen. If the lower endpoint of a range is omitted, then it is assumed to be zero. If the upper endpoint of a range is omitted, then it is assumed to be the highest CPU known to the operating system. To specify all processors type `-` or `0-`.

Applies to

Microsoft Windows and Linux.

Remarks

Setting this option is equivalent to executing the ProcessorAffinity system procedure.

The -gta option allows you to specify which logical processors are used by the database server.

The database server might not use all of the specified logical processors in the following cases:

- If one or more of the specified logical processors does not exist, or is offline.
- If the license does not allow it.

If the set of logical processors specified by the -gta option exceeds the license limits, the server uses the lowest-numbered logical processors specified by the -gta option up to, but not exceeding, the license limits. The database server does not choose logical processors in the order listed by the -gta option. The server does not attempt to maximize concurrency within the license limit and the specified -gta option.

If the set of logical processors specified by the -gta option does not match any online processors, the server acts as if the -gta option was not specified and uses up to the licensed number of logical processors, starting at processor 0.

The -gta option cannot be used with the -gt or -gtc options.

While the database server is running, you can change which logical processors it uses with the ProcessorAffinity option of the sa_server_option system procedure.

Example

The following example requires the database server to use some or all of the logical processors 3, 5, 6, 7, and/or 9:

```
dbsrv17 -gta 3,5-7,9
```

Related Information

Editions and Licensing
-gt Database Server Option [page 446]
-gtc Database Server Option [page 449]
sa_cpu_topology System Procedure
sa_server_option System Procedure

1.4.1.42 -gtc Database Server Option

Controls the maximum processor concurrency that the database server allows.

Syntax

```
dbsrv17 -gtc logical-processors-to-use ...
```

Applies to

Windows, Linux, and Solaris operating systems running on Intel-compatible x86 and x64 platforms.

Remarks

When you start the database server, the number of physical and logical processors detected by the database server appears in the database server messages window.

Physical processors are sometimes referred to as packages or dies, and are the CPUs of the computer. Additional logical processors exist when the physical processors support hyperthreading or are themselves configured as multiprocessors (often referred to as multi-core processors). The operating system schedules threads on logical processors.

The -gtc option allows you to specify the number of logical processors that can be used by the database server. Its effect is to limit the number of database server threads that are created at server startup. This limits the number of active database server tasks that can execute concurrently at any one time. By default, the number of threads created is 1 + the number of logical processors on all licensed physical processors.

By default, the database server allows concurrent use of all logical processors (cores or hyperthreads) on each licensed physical processor.

For example, when using a one-CPU license on a two-CPU system where each CPU contains four cores with two threads per core, the network database server permits eight threads to run concurrently on one CPU and zero threads on the other. If the -gtc option is specified, and the number of logical processors to be used is less than the total available for the number of physical processors that are licensed, then the database server allocates logical processors based on round-robin assignment. Specifying 1 for the -gtc option implicitly disables intra-query parallelism (parallel processing of individual queries). Intra-query parallelism can also be explicitly limited or disabled using the max_query_tasks option.

The -gtc option cannot be used with the -gta option.

The personal database server is limited to four cores on one CPU.

Example

Consider the following examples for a Windows-based SMP computer. In each case, assume a 4-processor system with two cores on each physical processor for a total of eight logical processors. The physical

processors are identified with letters and the logical processors (cores in this case) are identified with numbers. This 4-processor system therefore has processing units A0, A1, B0, B1, C0, C1, D0, and D1.

Scenario	Network database server settings
A single CPU license or -gt 1 specified	• -gt 1 • -gtc 2 • -gnh 20 Threads can execute on A0 and A1.
No licensing restrictions on the CPU with -gtc 5 specified	• -gt 4 • -gtc 5 • -gnh 20 Threads can execute on A0, A1, B0, C0, and D0.
A database server with a three CPU license and -gtc 5 specified	• -gt 3 • -gtc 5 • -gnh 20 Threads can execute on A0, A1, B0, B1, and C0.
No licensing restrictions on the CPU with -gtc 1 specified	• -gt 4 • -gtc 1 • -gnh 20 Threads can execute only on A0.

Related Information

Threading [page 344]
max_query_tasks Option [page 767]
-gnh Database Server Option [page 439]
-gt Database Server Option [page 446]
-gta Database Server Option [page 447]
sa_cpu_topology System Procedure
sa_server_option System Procedure

1.4.1.43 -gtp Database Server Option

Turns topology-aware scheduling on or off. This option is only useful on multiprocessor systems.

Syntax

```
dbsrv17 -gtp { 0 | 1 } ...
```

Allowed Values

0

Turns topology-aware scheduling off.

1

Turns topology-aware scheduling on.

Default

1

Applies to

Microsoft Windows and Linux.

Remarks

When topology-aware scheduling is On, tasks are scheduled to use a single thread on each available core before attempting to use a second thread on any core.

Related Information

Editions and Licensing
Threading [page 344]
-gnh Database Server Option [page 439]
-gtc Database Server Option [page 449]
-gta Database Server Option [page 447]
sa_cpu_topology System Procedure
sa_server_option System Procedure

1.4.1.44 -gu Database Server Option

Sets the privilege required for executing database file administration statements such as for creating or dropping databases.

Syntax

```
dbsrv17 -gu { DBA | ALL | NONE | utility_db } ...
```

Allowed Values

-gu option	Effect	Applies to
DBA	Only users with the SERVER OPERATOR system privilege can execute file administration statements	Any database including utility database
ALL	*This option is deprecated.* Anyone can execute file administration statements.	Any database including utility database
NONE	Executing file administration statements is not allowed.	Any database including utility database
utility_db	Only the users who can connect to the utility database can execute file administration statements	Only the utility database

Default

DBA

Applies to

All operating systems.

Remarks

Restricts the users who can execute the following database file administration statements:

- ALTER DATABASE dbfile ALTER TRANSACTION LOG
- CREATE DATABASE statement

- CREATE DECRYPTED DATABASE statement
- CREATE DECRYPTED FILE statement
- CREATE ENCRYPTED DATABASE statement
- CREATE ENCRYPTED FILE statement
- DROP DATABASE statement
- RESTORE DATABASE statement.

When utility_db is specified, these statements can only be run from the utility database. When DBA is specified, these statements can only be run by a user with the SERVER OPERATOR system privilege. When none is specified, no user can execute these statements.

Example

To prevent the use of the file administration statements, start the database server using the none privilege level of the -gu option. The following command starts a database server and names it TestSrv. It loads the `mytestdb.db` database, but prevents anyone from using that server to create or delete a database, or execute any other file administration statement regardless of their resource creation rights, or whether they can load and connect to the utility database.

```
dbsrv17 -n TestSrv -gu none c:\mytestdb.db
```

To permit only the users knowing the utility database password to execute file administration statements, start the server by running the following command.

```
dbsrv17 -n TestSrv -su passwd -gu utility_db
```

The following command starts Interactive SQL as a client application, connects to the server named TestSrv, loads the utility database, and connects the user.

```
dbisql -c "UID=DBA;PWD=passwd;DBN=utility_db;Host=host1;Server=TestSrv"
```

Having executed the above command successfully, the user connects to the utility database, and can execute file administration statements.

Related Information

The Utility Database (utility_db) [page 309]
CREATE DATABASE Statement
DROP DATABASE Statement
-su Database Server Option [page 502]

1.4.1.45 -im Database Server Option

Runs the database server in memory, reducing or eliminating writes to disk. With the exception of the validation option, in-memory mode requires a separate license.

Syntax

```
dbsrv17 -im { NONE | C | NW | V } ...
```

Allowed Values

NONE

In-memory mode is not used. This is the server default.

C (Checkpoint only)

When running in checkpoint-only mode, the database server does not use a transaction log, so you cannot recover to the most recent committed transaction. However, because the checkpoint log is enabled, the database can be recovered to the most recent checkpoint. Normally when you run a database without a transaction log, the database server still performs a checkpoint on a commit, which affects performance. However, when you run the database server in checkpoint-only mode, the database server does not perform a checkpoint after each commit.

Use this mode in applications where increased performance is desirable and the loss of committed transactions after the most recent checkpoint is acceptable.

The following restrictions apply when running in checkpoint-only mode:

1. There is no transaction log.
2. There is no temporary file.
3. Checkpoints are allowed both on demand and at the database server normal checkpoint frequency.
4. Dirty pages are flushed to disk only on checkpoint.

NW (Never write)

When running in never-write mode, committed transactions are not written to the database file on disk. All changes are lost if the database is shut down or crashes, so database files are always left in their original state. You can create and use new dbspaces, but they are not written to disk. Making a backup in never-write mode is not useful because any changes to the system dbspace are not written to the file.

The following restrictions apply when running in never-write mode:

1. Requests to extend or create new dbspaces are allowed, but the changes are not reflected in the database files.
2. There is no transaction log.
3. There is no checkpoint log.
4. There is no temporary file.
5. Dirty database pages are never flushed to disk.
6. The original database file is never modified.

V (Validation)

Use this mode to validate a backup copy of the database without modifying it. This mode does not require a separate license.

When running in validation mode, database files and transaction logs are always left in their original state: recovery operations are performed but are not written to the database file on disk, and all changes are lost if the database is shut down or crashes.

Databases are not shut down after recovery even when flags such as -a, -ad, or -f are specified.

The following criteria apply when running in validation mode:

1. Requests to extend, create, or use new dbspaces are not allowed.
2. Requests to modify the database are not allowed.
3. The database server continues to run.
4. Client connections are read-only.
5. There is no transaction log.
6. There is no checkpoint log.
7. There is a temporary file.
8. Dirty database pages are never flushed to disk.
9. The original database file and transaction logs are never modified.

Applies to

All operating systems.

With the exception of the validation option, in-memory mode requires a separate license.

Remarks

This feature is most useful on systems with a large amount of available memory, typically enough to hold all the database files within the cache.

In never-write mode (-im nw), because changes are never written to the original database files, if a persistent copy of current database contents is required, you must use the dbunload utility or the UNLOAD TABLE statement. You can also use SQL queries to retrieve the changes, but you must then manually write these changes to the database file.

The performance benefits gained from in-memory mode depend on the application workload and the speed of the I/O subsystem. The largest performance gains are seen in applications that insert or update large amounts of data, and in applications that commit and checkpoint frequently.

Often, performance of the in-memory modes is as good as, or better than, the performance of using transactional global temporary tables. The smallest performance improvement may be seen with applications that predominately query the database. In general, when using in-memory mode, the best performance can be achieved by pre-growing the cache to an amount large enough to hold the full expected contents of the database files. This eliminates much of the overhead involved in growing the cache in increments while the application is running.

⚠ Caution

Because pages are not flushed from cache in never-write or validation mode, it is possible to exhaust the available cache if the amount of data in the database grows too large. When this happens, the database server issues an error and stops processing requests. For this reason, never-write mode should be used with caution, and always with a cache large enough to hold the expected complete working set of pages that an application may use.

Validation mode should only be used with a cache large enough to hold all pages modified by the recovery.

Since checkpoints continue to occur in checkpoint-only mode, there is a reduced risk of the database server running out of available cache as compared to the never-write or validation modes.

For the LOAD TABLE and some ALTER TABLE statements, the checkpoint log is used to partially reverse the effects of a failure or to recover from an error. In never-write mode, a checkpoint log is not created and you cannot partially reverse the effects of some statements if they fail or an error occurs. Incorrect or incomplete data could remain in tables.

Related Information

Checkpoint Logs [page 307]
-c Database Server Option [page 400]

1.4.1.46 -k Database Server Option

Controls the collection of Windows Performance Monitor statistics and statement performance summary statistics.

Syntax

```
dbsrv17 -k ...
```

Default

Windows Performance Monitor statistics and statement performance summaries are collected.

Applies to

All operating systems.

Remarks

If you specify -k when you start the database server, then the database server does not collect statistics for the Windows Performance Monitor. The -k option does not affect the collection of column statistics used by the query optimizer. You can also change the settings for the collection of Windows Performance Monitor statistics and statement performance data using the sa_server_option system procedure.

Additionally, this option disables statement performance summary collection. To disable statement performance summary collection on a single database, set the CollectStmtPerfStats option of the sa_db_option system procedure. Setting this option requires MONITOR privileges.

Statement performance data is collected only if both the database server and the database have data collection enabled.

This option should only be used in situations where the database server is running on a multi-processor computer where it can be shown by testing to improve performance. For most workloads, the benefit will be negligible, so use of this option is not recommended. When you disable the performance counters, this information is not available for analyzing performance problems.

Related Information

Tip: Identify the Cause of Slow Statements [page 1447]
sa_server_option System Procedure
-ks Database Server Option [page 462]
-ksc Database Server Option [page 463]
-ksd Database Server Option [page 463]
sa_server_option System Procedure

1.4.1.47 -kl Database Server Option

Specifies the file name of the Kerberos GSS-API library (or shared object on UNIX and Linux) and enables Kerberos authenticated connections to the database server.

Syntax

```
dbsrv17 -kl GSS-API-library-file ...
```

Applies to

All operating systems.

Remarks

This option specifies the location and name of the Kerberos GSS-API. This option is only required if the Kerberos client uses a different file name for the Kerberos GSS-API library than the default, or if there are multiple GSS-API libraries installed on the computer running the database server. A Kerberos client must already be installed and configured, and SSPI cannot be used by the database server.

Specifying this option enables Kerberos authentication to the database server.

Example

The following command starts a database server that uses the `libgssapi_krb5.so` shared object for Kerberos authentication.

```
dbsrv17 -kl libgssapi_krb5.so -n my_server_princ /opt/myapp/kerberos.db
```

Related Information

Kerberos User Authentication [page 159]
-kr Database Server Option (Deprecated) [page 459]
-krb Database Server Option [page 461]
Kerberos (KRB) Connection Parameter [page 92]

1.4.1.48 -kp Database Server Option

Specifies the Kerberos server principal and enables Kerberos authenticated connections to the database server.

Syntax

```
dbsrv17 -kp server-principal ...
```

Applies to

All operating systems.

Remarks

This option specifies the Kerberos server principal used by the database server and enables Kerberos authentication. By default, the principal used by the database server for Kerberos authentication is `server-name@default-realm`, where `default-realm` is the default realm configured for the Kerberos client. Specify this option to use a different server principal. The industry format for Kerberos server principals is `server-name/hostname@realm`.

If OpenClient or jConnect Kerberos authenticated connections are made to the database server, the server principal must also be specified by the application. See SERVICE_PRINCIPAL_NAME in the jConnect documentation.

The -kr option cannot be specified if the -kp option is specified.

Example

The following command starts a database server that accepts Kerberos logins and uses the server principal myserver/mycomputer.example.com@EXAMPLE.COM for authentication.

```
dbsrv17 -kp myserver/mycomputer.example.com@EXAMPLE.COM -n myServerName C:
\kerberos.db
```

Related Information

Kerberos User Authentication [page 159]
-kl Database Server Option [page 457]
-krb Database Server Option [page 461]
-kr Database Server Option (Deprecated) [page 459]
Kerberos (KRB) Connection Parameter [page 92]

1.4.1.49 -kr Database Server Option (Deprecated)

Specifies the realm of the Kerberos server principal and enables Kerberos authenticated connections to the database server.

> **i** Note
>
> The use of the -kr option is deprecated. Use the -kp option to specify the Kerberos server principal. If you specify -kp, the server principal must have been extracted to the Kerberos keytab file on the computer running the database server.

Syntax

```
dbsrv17 -kr server-realm ...
```

Applies to

All operating systems.

Remarks

This option specifies the realm of the Kerberos server principal. Normally, the principal used by the database server for Kerberos authentication is `server-name@default-realm`, where `default-realm` is the default realm configured for the Kerberos client. Use this option if you want the server principal to use a different realm than the default realm, in which case the server principal used is `server-name@server-realm`.

Specifying this option enables Kerberos authentication to the database server.

The -kr option cannot be specified if the -kp option is specified.

Example

The following command starts a database server that accepts Kerberos logins and uses the principal my_server_princ@MYREALM for authentication.

```
dbsrv17 -kr MYREALM -n my_server_princ C:\kerberos.db
```

Related Information

Kerberos User Authentication [page 159]
-kl Database Server Option [page 457]
-kp Database Server Option [page 458]
-krb Database Server Option [page 461]
Kerberos (KRB) Connection Parameter [page 92]

1.4.1.50 -krb Database Server Option

Enables Kerberos-authenticated connections to the database server.

Syntax

```
dbsrv17 -krb ...
```

Applies to

All operating systems.

Remarks

This option enables Kerberos user authentication to the database server. You must specify one or more of the -krb, -kl, and -kr options for the database server to authenticate clients using Kerberos.

Before you can use Kerberos user authentication, a Kerberos client must already be installed and configured on both the client and database server computers. Additionally, the principal `server-name@REALM` must already exist in the Kerberos KDC, and the keytab for the principal `server-name@REALM` must already have been securely extracted to the keytab file on the database server computer. The database server will not start if the -krb option is specified, but this setup has not been performed.

Note

The database server name cannot contain any of the following characters: /, \, or @, and database server names with multibyte characters cannot be used with Kerberos.

The login_mode database option must be set to allow Kerberos logins, and Kerberos client principals must be mapped to database user IDs using the GRANT KERBEROS LOGIN statement.

Example

For a Kerberos principal for the database server named my_server_princ@MYREALM, the following command starts a database server named my_server_princ.

```
dbsrv17 -krb -n my_server_princ C:\kerberos.db
```

Related Information

Kerberos User Authentication [page 159]
-kl Database Server Option [page 457]
-kr Database Server Option (Deprecated) [page 459]
Kerberos (KRB) Connection Parameter [page 92]
login_mode Option [page 752]

1.4.1.51 -ks Database Server Option

Disables the creation of shared memory that the Windows Performance Monitor uses to collect counter values from the database server.

Syntax

```
dbsrv17 -ks 0 ...
```

Applies to

Windows.

Remarks

When you specify this option, the Windows Performance Monitor does not show any database server, database, or connection statistics for the current database server.

Related Information

Tip: Identify the Cause of Slow Statements [page 1447]
-k Database Server Option [page 456]
-ksc Database Server Option [page 463]
-ksd Database Server Option [page 463]

1.4.1.52 -ksc Database Server Option

Specifies the maximum number of connections that the Windows Performance Monitor can monitor.

Syntax

```
dbsrv17 -ksc integer ...
```

Default

10

Applies to

Windows.

Related Information

Tip: Identify the Cause of Slow Statements [page 1447]
-k Database Server Option [page 456]
-ks Database Server Option [page 462]
-ksd Database Server Option [page 463]

1.4.1.53 -ksd Database Server Option

Specifies the maximum number of databases that the Windows Performance Monitor can monitor.

Syntax

```
dbsrv17 -ksd integer ...
```

Default

2

Applies to

Windows.

Related Information

Tip: Identify the Cause of Slow Statements [page 1447]
-k Database Server Option [page 456]
-ks Database Server Option [page 462]
-ksc Database Server Option [page 463]

1.4.1.54 -lt Databases Server Option

Sets the number of lock tables for each database.

Syntax

```
dbsrv17 -lt n
```

Default

0

Allowed Values

0 The SQL Anywhere server/engine decides a suitable value to use.

n Any number that is a power of two between 1 and 128. The value will be rounded up to the nearest power of two less than or equal to 128, if needed.

Applies to

Applies to all operating systems.

Remarks

At startup, SQL Anywhere selects how many lock tables to create based on how large the server appears to be. In general, more lock tables allow more simultaneous transactions, at a cost of slightly higher memory usage. On very large systems with hundreds of concurrent transactions, the selected number of lock tables may be too low. This option allows users to see how many lock tables were selected, to see if there are too few lock tables for a given workload, and to override the automatically selected number of lock tables if necessary.

1.4.1.55 -m Database Server Option

Truncates the transaction log when a checkpoint is done.

Syntax

```
dbsrv17 -m ...
```

Applies to

All operating systems.

Remarks

This option truncates the transaction log when a checkpoint is done, either at shutdown or as a result of a checkpoint scheduled by the server.

> ⚠ Caution
>
> When this option is selected, there is no protection against media failure on the device that contains the database files.

This option provides a way to automatically limit the growth of the transaction log. Checkpoint frequency is still controlled by the checkpoint_time and recovery_time options (which you can also set on the command line).

The -m option is useful for limiting the size of the transaction log in situations where high volume transactions requiring fast response times are being processed, and the contents of the transaction log aren't being relied upon for recovery or replication. The -m option provides an alternative to operating without a transaction log at all, in which case a checkpoint would be required following each COMMIT and performance would suffer as a result. When the -m option is specified, there is no protection against media failure on the device that contains the database files. Other alternatives for managing the transaction log (for example, using the BACKUP DATABASE statement and events) should be considered before using the -m option.

To avoid database file fragmentation, place the transaction log on a separate device or partition from the database itself.

When this option is used, no operations can proceed while a checkpoint is in progress.

> ⚠ Caution
>
> Do not use the -m option with databases that are being replicated or synchronized.

Related Information

The Transaction Log [page 292]
Checkpoint Logs [page 307]
-m Database Option [page 553]
Transaction Log Utility (dblog) [page 1231]
checkpoint_time Option [page 700]
recovery_time Option [page 806]

1.4.1.56 -md Database Server Option

Controls the crash dump type.

> Syntax

```
dbsrv17 -md ...
```

Applies to

All operating systems. Only NORMAL is supported on UNIX platforms.

Allowed Values

NORMAL

Enables the generation of mini crash dumps.

FULL

Enables the generation of full crash dumps

Remarks

This option should be set to NORMAL unless requested by technical support as a FULL crash dump can be very large.

Once the database server is started, you can change the crash dump type by using the sa_server_option system procedure.

You can find the current value of MiniDumpType by running the following query:

```
SELECT PROPERTY( 'MiniDumpType' );
```

Related Information

sa_server_option System Procedure

1.4.1.57 -n Database Server Option

Sets the name of the database server.

Syntax

```
dbsrv17 -n server-name database-filename ...
```

Default

The name of the first database file (with the path and extension removed) that is started on the database server.

Applies to

All operating systems.

Remarks

When a database server starts, it attempts to become the default database server on that computer. The first database server to start when there is no default server becomes the default database server. Shared memory

connection attempts on that computer that do not explicitly specify a database server name connect to the default server.

> **Note**
>
> Use the -xd option for database servers being used by deployed applications, and that all clients explicitly specify the name of the database server to which they should connect by using the ServerName (Server) connection parameter. This ensures that the database connects to the correct database server when a computer is running multiple database servers.

There is no character set conversion performed on the server name. If the client character set and the database server character set are different, using extended characters in the server name can cause the server to not be found. If your clients and servers are running on different operating systems or locales, you should use 7-bit ASCII characters in the server name.

Database server names must be valid identifiers. Long database server names are truncated to different lengths depending on the protocol. Database server names cannot:

- begin with white space, single quotes, or double quotes
- end with white space
- contain semicolons, forward slashes (/), or backslashes (\)
- be longer than 250 bytes
- contain spaces when they are running on UNIX or Linux

> **Note**
>
> On Windows and UNIX/Linux, version 9.0.2 and earlier clients cannot connect to version 10.0.0 and later database servers with names longer than the following lengths:
>
> - 40 bytes for Windows shared memory
> - 31 bytes for UNIX or Linux shared memory
> - 40 bytes for TCP/IP

The server name specifies the name to be used in the ServerName (Server) connection parameter of client application connection strings or profiles. With shared memory, unless -xd is specified, a default database server is used if a server name is not specified and there is at least one database server running on the computer.

Running multiple database servers with the same name is not recommended.

> **Note**
>
> There are two -n options. The -n option is positional. If it appears before any database file names, it is a server option and names the server. If it appears after a database file name, it is a database option and names the database.

Example

In the following example, the database server is started on the file *%SQLANYSAMP17%\demo.db*. The server name is DemoServer.

```
dbsrv17 -n DemoServer %SQLANYSAMP17%\demo.db
```

If no -n option is specified, the name of the server is `demo`.

The following example uses the database server option -n and the database option -n. It names the database server SERV and the database DATA.

```
dbsrv17 -n SERV sales.db -n DATA
```

Related Information

Database Server Names and Database Names [page 334]
Connection Strings and Character Sets [page 607]
Identifiers
ServerName (Server) Connection Parameter [page 110]
-xd Database Server Option [page 526]
-n Database Option [page 554]

1.4.1.58 -ncs Database Server Option

Enables sending status events to the SAP Solution Manager using the NCS (Native Component Supportability) library.

Syntax

```
dbsrv17 -ncs ...
```

Applies to

Windows and x64 Linux.

Remarks

The location of the NCS configuration file is determined from the setting of the NCS_CONFIG environment variable. If the environment variable is not set, the NCS library will look in the default folder (`/usr/sap/ncs/`

`config` for Linux and `c:\usr\sap\ncs\config` for Windows), followed by the current working directory. The configuration file contains the settings required to connect to SAP Solution Manager.

Example

```
dbsrv17 -ncs
```

Related Information

Solution Manager [page 1370]
-ncsd Database Server Option [page 470]

1.4.1.59 -ncsd Database Server Option

Enables sending status events to the SAP Solution Manager using the NCS (Native Component Supportability) library, and specifies the location of the NCS configuration file to use.

Syntax

```
dbsrv17 -ncsd filename ...
```

Applies to

Windows and x64 Linux.

Remarks

`filename` specifies the path and file name of the NCS configuration file. The configuration file contains the settings required to connect to SAP Solution Manager.

Example

```
dbsrv17 -ncsd c:\ncs\ncs.config
```

Related Information

Solution Manager [page 1370]
-ncs Database Server Option [page 469]

1.4.1.60 -o Database Server Option

Prints all database server messages to the database server message log file.

Syntax

```
dbsrv17 -o filename ...
```

Applies to

All operating systems.

Remarks

Print all database server messages, including informational messages, errors, warnings, and MESSAGE statement output, to the specified file, and to the database server messages window. If you specify the -qi option with -o, all messages appear only in the database server message log file.

Do not end the file name with `.log` because this can create problems for utilities that perform operations using the transaction log.

To find the name of the database server message log file, execute the following statement:

```
SELECT PROPERTY ( 'ConsoleLogFile' );
```

Related Information

Database Server Logging [page 336]
-oe Database Server Option [page 472]
-on Database Server Option [page 473]
-os Database Server Option [page 475]
-ot Database Server Option [page 476]
sa_server_option System Procedure
-qi Database Server Option [page 483]

1.4.1.61 -oe Database Server Option

Specifies a file name to log startup errors, fatal errors, and assertions.

Syntax

```
dbsrv17 -oe filename ...
```

Applies to

All operating systems.

Remarks

Each line in the output log file is prefixed with the date and time. Startup errors include such errors as:

- Couldn't open/read database file: `database-filename`.
- A database server with that name has already started.

Fatal errors and assertions are logged to the Windows Application Event Log or the Unix system log, regardless of whether -oe is specified.

Do not end the file name with `.log` because this can create problems for utilities that perform operations using the transaction log.

Related Information

-o Database Server Option [page 471]
-on Database Server Option [page 473]
-os Database Server Option [page 475]
-ot Database Server Option [page 476]
-qi Database Server Option [page 483]

1.4.1.62 -on Database Server Option

Specifies a maximum size for the database server message log, after which the file is renamed with the extension `.old` and a new file is started.

Syntax

```
dbsrv17 -on size[ k | m | g ] ...
```

Applies to

All operating systems.

Remarks

The `size` is the maximum file size for the database server message log, in bytes. Use k, m, or g to specify units of kilobytes, megabytes, or gigabytes respectively. The minimum size limit is 10 KB. By default, there is no maximum size limit.

When the database server message log reaches the specified size, the database server renames the file with the extension `.old`, and starts a new file with the original name.

Note

If the `.old` database server message log file already exists, it is overwritten. To avoid losing old database server message log files, use the -os option instead.

This option cannot be used with the -os option.

Do not end the database server message log file name with `.log` because this can create problems for utilities that perform operations using the transaction log.

Related Information

Database Server Logging [page 336]
-o Database Server Option [page 471]
-oe Database Server Option [page 472]
-os Database Server Option [page 475]
-ot Database Server Option [page 476]
sa_server_option System Procedure

1.4.1.63 -orp Database Server Option

Resets the password of the specified user.

Syntax

```
dbsrv17 -orp { UserID | UID }=user-ID;{ NewPassword | NEWPWD }=new-password;
{ AuthUserID | AUTHUID }=auth-user-ID;{ AuthPassword | AUTHPWD }=auth-password"
database-file
```

On UNIX and Linux, quotation marks are required for the -orp value when semicolons are used.

Applies to

All operating systems.

Permissions

SYS_OFFLINE_RESET_PASSWORD_ROLE system role

SERVER OPERATOR system privilege

Allowed Values

user-ID

The user whose password is being reset.

new-password

The new password for `user-ID`.

auth-user-ID

The user who has been granted the SYS_OFFLINE_RESET_PASSWORD_ROLE system role by the role administrator.

auth-password

The password for `auth-user-ID`.

database-file

The database that contains the user whose password you are resetting.

Remarks

You can only specify one database file with -orp.

The `user-ID` user (normally the DBA) whose password you are resetting must have a password. Password reset is not allowed for users without a valid password.

The new password must have a length of at least six characters.

You cannot specify any other database server options with the -orp option, with the exception of -o, -ep, and -ek.

You must shut down the database server before restarting it with the -orp option. When you specify -orp, the database server starts and attempts to reset the password of the specified user. The database server shuts down immediately after the password reset is attempted. After the shutdown, a message appears indicating that the password reset was successful or showing the reason that the reset failed.

> **i Note**
>
> The database user that starts the database sever with the -orp option must have the SYS_OFFLINE_RESET_PASSWORD_ROLE system role.

Example

The password for the user `DBA` is lost. The user reset_user has the SYS_OFFLINE_RESET_PASSWORD_ROLE system role and resets the `DBA` user's password to be **`newpassword`**:

```
dbeng17 -orp "UID=DBA;NEWPWD=newpassword;AUTHUID=reset_user;AUTHPWD=sql456"
mydb.db
```

Related Information

System Roles [page 1528]
Super-users [page 1558]
Resetting the DBA Password (Command Line) [page 1637]

1.4.1.64 -os Database Server Option

Specifies a maximum size for the database server message log file, at which point the file is renamed.

> **‹≡› Syntax**
>
> ```
> dbsrv17 -os size[ k | m | g ] ...
> ```

Applies to

All operating systems.

Remarks

The `size` is the maximum file size for logging database server messages, in bytes. Use k, m, or g to specify units of kilobytes, megabytes, or gigabytes respectively. The minimum size limit is 10 KB. By default, there is no maximum size limit.

Before the database server logs output messages to the database server message log file, it checks the current file size. If the log message will make the file size exceed the specified size, the database server renames the database server message log file to `yymmddxx.slg`, where `yymmdd` represents the year, month, and day the file was created, and `xx` is a number that starts at 00 and continues incrementing.

This option allows you to identify old database server message log files that can be deleted to free up disk space.

This option cannot be used with the -on option.

Do not end the database server message log file name with `.log` because this can create problems for utilities that perform operations using the transaction log.

Related Information

Database Server Logging [page 336]
-o Database Server Option [page 471]
-oe Database Server Option [page 472]
-on Database Server Option [page 473]
-ot Database Server Option [page 476]

1.4.1.65 -ot Database Server Option

Truncates the database server message log file and appends output messages to it.

Syntax

```
dbsrv17 -ot logfile ...
```

Applies to

All operating systems.

Remarks

The functionality is the same as the -o option except the database server message log file is truncated before any messages are written to it. To find the name of the database server message log file, execute the following statement:

```
SELECT PROPERTY ( 'ConsoleLogFile' );
```

Do not end the database server message log file name with `.log` because this can create problems for utilities that perform operations using the transaction log.

Related Information

Database Server Logging [page 336]
-o Database Server Option [page 471]
-oe Database Server Option [page 472]
-on Database Server Option [page 473]
-os Database Server Option [page 475]

1.4.1.66 -p Database Server Option

Sets the maximum size of communication packets.

Syntax

```
dbsrv17 -p integer ...
```

Default

7300 bytes (all operating systems)

Applies to

All operating systems.

Remarks

The minimum value is 1000 bytes and the maximum value is 65535.

You can change the communication buffer size for a connection by setting the CommBufferSize (CBSIZE) connection parameter.

Related Information

-pc Database Server Option [page 478]
-pt Database Server Option [page 482]
CommBufferSize (CBSIZE) Connection Parameter [page 58]

1.4.1.67 -pc Database Server Option

Compresses all connections except for same-computer connections.

Syntax

```
dbsrv17 -pc ...
```

Applies to

All operating systems.

Network database server (dbsrv17) only.

Remarks

The packets sent between a client and database server can be compressed using the -pc option. Compressing a connection may improve performance under some circumstances. Large data transfers with highly compressible data tend to get the best compression rates. This option can be overridden for a particular client by specifying COMPRESS=NO in the client's connection parameters.

By default, connections are not compressed. Specifying the -pc option compresses all connections except same-computer connections, web services connections, and TDS connections. TDS connections (including jConnect) do not support communication compression.

Same-computer connections over any communication link are not compressed, even if the -pc option or COMPRESS=YES connection parameter is used.

Related Information

Communication Compression Settings [page 185]
-p Database Server Option [page 477]
-pt Database Server Option [page 482]
Compress (COMP) Connection Parameter [page 61]

1.4.1.68 -pf Database Server Option

Writes the process ID into the specified file.

Syntax

```
dbsrv17 -pf filename ...
```

Applies to

All operating systems.

Remarks

The specified file is created when the process starts and deleted when the process exits. If the file exists at creation time, then it is overwritten if it contains a single number in its first line and that number does not refer to a process running on the system.

The process must have sufficient permissions to write the specified file. If the file cannot be created, then the process exists with an error message.

Example

```
dbsrv17 "%SQLANYSAMP17%\demo.db" -pf processid.txt
```

Related Information

Broadcast Repeater Utility (dbns17) [page 1128]
SQL Anywhere Database Server Executable (dbsrv17, dbeng17) [page 380]

1.4.1.69 -phl Database Server Option

Sets history tracking for specified database server properties.

Syntax

```
dbsrv17 -phl { OFF | ON | NONE | property-name, ... }
```

On UNIX and Linux, quotation marks are required when more than one parameter is supplied:

Default

ON

Applies to

All operating systems.

Allowed Values

ON When PropertyHistoryList is turned on, a default list of properties is tracked. The default is ON.
OFF When PropertyHistoryList is turned off property tracking is disabled for the database server.
NONE Setting this property to NONE enables property tracking, but no properties are tracked by the database server. Only properties requested by databases are tracked.
Comma-delimited list of database server properties

Specify a comma-delimited list of database server properties to track.

Remarks

The -phl database server option allows you to specify what database server properties, if any, are to be tracked by the server.. If this option is not set or is set to ON, then the database server uses the default setting of the PropertyHistoryList database server property.

Related Information

sp_property_history System Procedure
PROPERTY_IS_TRACKABLE Function [System]
-sf Database Server Option [page 493]
sa_server_option System Procedure
sa_db_option System Procedure
-phs Database Server Option [page 481]
List of Database Server Properties [page 900]
List of Database Properties [page 917]
System Privileges [page 1577]
Viewing All Trackable Database Server Property Values [page 934]

1.4.1.70 -phs Database Server Option

Sets the maximum amount of memory to use for tracking property history, in allotted time or bytes.

Syntax

```
dbsrv17 -phs { [hh:]mm:ss | size [ k | m ] | MAX | DEFAULT } ...
```

Default

DEFAULT

Applies to

All operating systems.

Remarks

The -phs database server option specifies the maximum size allowed for tracking property history. The size is measured in allotted time per property or bytes of memory. The `size` parameter also increases the initial cache size of the database server to avoid having to grow the cache during start up. This increase in initial cache size prevents long start up times. If this option is not set, then the database server uses the *DEFAULT* setting of the PropertyHistorySize database server option.

If you specify a value for the -phs database server option, but the -phl database server option is set to OFF, then no property history is collected. If you set the -phs database database server property to 0 or to a time-based value, then no space is allocated for property history tracking.

Specifying *MAX* sets the maximum size for tracking property history to 2% of the cache, or 256 MB, whichever is smaller.

Specifying *DEFAULT* sets the size for tracking property history to either 10 minutes or to a maximum of 2% of the cache.

> **i** Note
>
> If the specified size does not allow the database server to store the minimum amount of data for each tracked database server property (1024 bytes, or approximately 30 seconds), then an error is returned.

Related Information

sp_property_history System Procedure
PROPERTY_IS_TRACKABLE Function [System]
-sf Database Server Option [page 493]
sa_server_option System Procedure
sa_db_option System Procedure
-phl Database Server Option [page 480]
List of Database Server Properties [page 900]
List of Database Properties [page 917]
System Privileges [page 1577]

1.4.1.71 -pt Database Server Option

Increases or decreases the size limit at which TCP/IP packets are compressed.

> **‹≡›** Syntax

```
dbsrv17 -pt size ...
```

Default

120 bytes

Applies to

All operating systems.

Network database server (dbsrv17) only.

Remarks

This parameter takes an integer value representing the minimum byte-size of packets to be compressed. Values less than 80 are not recommended.

Under some circumstances, changing the compression threshold can help performance of a compressed connection by allowing you to compress packets only when compression will increase the speed at which the packets are transferred. The default setting should be appropriate for most cases.

If both client and server specify different compression threshold settings, the client setting applies.

Related Information

Communication Compression Settings [page 185]
-p Database Server Option [page 477]
-pc Database Server Option [page 478]
CompressionThreshold (COMPTH) Connection Parameter [page 63]

1.4.1.72 -qi Database Server Option

Controls whether database server system tray icon and database server messages window appear.

Syntax

```
dbsrv17 -qi ...
```

Applies to

Windows.

Remarks

This option leaves no visual indication that the server is running, other than possible startup error windows. You can use either (or both) the -o or -oe log files to diagnose errors.

Related Information

-qn Database Server Option [page 484]
-qp Database Server Option [page 485]
-qs Database Server Option [page 486]
-qw Database Server Option [page 487]
-o Database Server Option [page 471]
-oe Database Server Option [page 472]

1.4.1.73 -qn Database Server Option

Specifies that the database server messages window is not minimized on startup.

Syntax

```
dbsrv17 -qn ...
```

Applies to

Windows.

Linux (if X window server is used).

Remarks

By default, the database server messages window automatically minimizes once database server startup completes. When this option is specified, the database server messages window does not minimize after the database server starts.

The database server messages window may appear in the background if an application automatically starts the database server when the application is not active and -qn is specified.

On Linux, you must specify the -ux option (use X window server) with the -qn option.

Example

The following command starts the database server on Linux or Solaris, displays the database server messages window, and does not minimize the database server messages window once the database server is started:

```
dbsrv17 -ux -qn sample.db
```

Related Information

-ux Database Server Option [page 519]
-qi Database Server Option [page 483]
-qp Database Server Option [page 485]
-qs Database Server Option [page 486]
-qw Database Server Option [page 487]

1.4.1.74 -qp Database Server Option

Specifies that messages about performance do not appear in the database server messages window.

Syntax

```
dbsrv17 -qp ...
```

Applies to

All operating systems.

Remarks

Do not display messages about performance in the database server messages window. Messages that are suppressed include the following:

- No unique index or primary key for table '`table-name`'
- Database file "`mydatabase.db`" consists of `nnn` fragments

Related Information

-qi Database Server Option [page 483]
-qn Database Server Option [page 484]
-qs Database Server Option [page 486]
-qw Database Server Option [page 487]

1.4.1.75 -qs Database Server Option

Suppresses startup error windows.

Syntax

```
dbsrv17 -qs ...
```

Applies to

Windows.

Remarks

This option conceals startup error windows. Examples of startup errors include the database server not being able to open or read a database file or a database server not starting because another database server with the specified name is already running.

On Windows platforms, if the server is not being started automatically, these errors appear in a window and must be cleared before the server stops. These windows do not appear if the -qs option is used.

If there is an error loading the language DLL, the error is not logged to the -o or -oe logs, but rather to the Windows Application Event Log.

Usage errors are suppressed if -qs is on the command line, but not in @data expansion.

Related Information

-qi Database Server Option [page 483]
-qn Database Server Option [page 484]
-qp Database Server Option [page 485]
-qw Database Server Option [page 487]
-o Database Server Option [page 471]
-oe Database Server Option [page 472]

1.4.1.76 -qw Database Server Option

Specifies that the database server messages window does not appear.

Syntax

```
dbsrv17 -qw ...
```

Applies to

All operating systems.

Remarks

This option suppresses the database server messages window. On Windows platforms, the database server system tray icon is still visible. You can use either (or both) the -o or -oe log files to diagnose errors.

Related Information

-qi Database Server Option [page 483]
-qn Database Server Option [page 484]
-qp Database Server Option [page 485]
-qs Database Server Option [page 486]

1.4.1.77 -r Database Server Option

Forces all databases that start on the database server to be read-only.

No changes to the database(s) are allowed: the database server doesn't modify the database file(s) or transaction log files.

Syntax

```
dbsrv17 -r ...
```

Applies to

All operating systems.

Remarks

Opens all database files as read-only with the exception of the temporary file when the option is specified before any database names on the command line. If the -r option is specified after a database name, only that specific database is read-only. You can make changes on temporary tables, but ROLLBACK has no effect, since the transaction and rollback logs are disabled.

A database distributed on a CD-ROM device is an example of a database file that cannot be modified. You can use read-only mode to access this sort of database.

If you attempt to modify the database, for example with an INSERT or DELETE statement, a SQLSTATE_READ_ONLY_DATABASE error is returned.

Databases that require recovery cannot be started in read-only mode unless the database server is also running in in-memory validation mode. Database files created using an online backup cannot be started in read-only mode if there were any open transactions when the backup was started since these transactions would require recovery when the backup copy is started.

Databases with auditing turned on cannot be started in read-only mode.

If you are checking the validity of a backup database, then you should run the database in read-only mode so that it is not modified. Alternatively, you can start the database server in in-memory validation or never write mode.

Example

The following command starts two databases in read-only mode.

```
dbsrv17 -r database1.db database2.db
```

The following command starts only the first of two databases in read-only mode.

```
dbsrv17 database1.db -r database2.db
```

Related Information

Database Employment on Read-only Media
Special Modes [page 341]
Validating a Database (SQL Central) [page 990]
-r Database Option [page 556]
auditing Option [page 688]

1.4.1.78 -s Database Server Option

Sets the user ID for Syslog messages.

Syntax

```
dbsrv17 -s { none | user | daemon | localn } ...
```

Applies to

UNIX and Linux.

Remarks

Sets the system user ID used in messages to the Syslog facility. The default is user for database servers that are started in the foreground, and daemon for those that are run in the background (for example, started by dbspawn, started automatically by a client, or started with the -ud database server option).

A value of none prevents any Syslog messages from being logged. The local*n* argument allows you to use a facility identifier to redirect messages to a file. You can specify a number between 0 and 7, inclusive, for *n*. For more information, refer to the UNIX or Linux Syslog(3) man page.

Related Information

MESSAGE Statement

1.4.1.79 -sb Database Server Option

Specifies how the database server reacts to broadcasts.

Syntax

```
dbsrv17 -sb { 0 | 1 } ...
```

Applies to

All operating systems. TCP/IP only.

Remarks

The database server starts one or more UDP listeners so that it can respond to three types of broadcasts:

1. client connection broadcasts looking for this server.
2. database server enumeration broadcasts (such as those from the dblocate utility or the *Find Servers Wizard* of the administration tools *Connect* window).
3. broadcasts sent by another database server that is looking for any database servers with the same name.

For Embedded SQL connections, the dblocate utility and db_locate_servers function cause broadcast packets to be sent out on local networks in an attempt to find all database servers. The UDP listener within the database server then responds back to the sender with information about how to connect to the database server.

The -sb option controls the behavior of the UDP listeners within the database server.

When the -sb option is not specified, the database server responds to all three types of requests.

If you specify -sb 0, then the database server responds to (3) only. This forces clients to use a Host connection parameter or HOST protocol option when connecting to the database server. In addition if the server is using a port other than the default port (2638), then the client must also specify the server's port. Since the database server does not respond to database server enumeration broadcasts, the database server is not included in the output of dblocate.

If you specify -sb 1, then the database server responds to (1) and (3). Since the database server does not respond to database server enumeration broadcasts, the database server is not included in the output of dblocate.

Related Information

BroadcastListener (BLISTENER) Protocol Option (Server Side Only) [page 193]
Server Enumeration Utility (dblocate) [page 1200]

Host Connection Parameter [page 86]
Host (IP) Protocol Option (Client Side Only) [page 206]

1.4.1.80 -sbx Database Server Option

Sets the default disk sandbox settings for all databases started on the database server that do not have explicit disk sandbox settings.

Syntax

```
dbsrv17 -sbx{ + | - } ...
```

Default

Off

Applies to

All operating systems.

Allowed Values

-sbx+ Enable disk sandbox settings for all databases.
-sbx Enable disk sandbox settings for all databases.
-sbx- Disable disk sandbox settings for all databases.

Remarks

This option specifies the default disk sandbox settings for all databases started on the database server that do not have explicit disk sandbox settings. The -sbx database option, which explicitly specifies disk sandbox settings for a database, overrides the -sbx database server option.

Enabling this option restricts the read-write file operations for all databases running on the database server to the directory where the main database file is located and any subdirectories of this directory.

Change the default disk sandbox settings for the database server while the database server is running by using the DiskSandbox option in the sa_server_option system procedure. The sa_server_option system procedure

does not affect databases that have already been started on the database server or databases that have explicit disk sandbox settings.

Related Information

-sbx Database Option [page 559]
disk_sandbox Option [page 730]
sa_server_option System Procedure

1.4.1.81 -sclr Database Server Option

Specifies that the database server uses one CLR external environment for all databases running on the database server

Syntax

```
dbsrv17 -sclr { 35 | 40 | 45 } ...
```

Default

The database server uses one CLR external environment for each database.

Applies to

Windows operating systems.

Remarks

This option specifies whether or not the database server uses one CLR external environment for all databases on the database server. The option value indicates which version to start. You can also set the database server to use one CLR external environment for all databases by calling the sa_server_option system procedure and setting the SingleCLRInstanceVersion database server option to a CLR version.

Example

The following command starts a database server named myserver1, that uses one CLR external environment for all databases running on the database server.

```
dbsrv17 -n myserver1 -sclr 35
```

Related Information

sa_server_option System Procedure

1.4.1.82 -sf Database Server Option

Controls whether users have access to features for databases running on the current database server.

Unsecured features can be accessed by any user with the appropriate system role or privilege. A secured feature can only be accessed by a user that has been given the authorization to use a secured feature (by specifying an appropriate secured feature key) and who has the appropriate system role or privilege.

Syntax

```
dbsrv17 -sf feature-list ...
```

```
feature-list :
feature-name | feature-set[,[-]feature-name | feature-set] ...
```

Parameters

none

Specifies that no features are secured.

manage_server

This feature set prevents users from accessing all database server-related features. This set consists of the following features:

processor_affinity

Prevents users from changing the processor affinity (the number of logical processors being used) of the database server.

manage_cockpitdb Prevents users from enabling or disabling the Cockpit, or from changing the default file for the Cockpit. That is, manage_cockpitdb prevents users from executing sa_server_option with the CockpitDB option.

manage_listeners

Prevents users from starting or stopping a connection listener by using the sp_start_listener or sp_stop_listener system procedure.

manage_property_history Prevents users from enabling and configuring the tracking of database server property values.

manage_security

This feature set prevents users from accessing features that allow the management of database server security. By default, these features are secured.

manage_features

Prevents users from modifying the list of features that can be secured on the database server.

manage_keys

Prevents the creation, modification, deletion, or listing of secured feature keys.

A user that has access to the manage_keys feature but not the manage_features feature cannot define a key with more features than those assigned to the user.

manage_disk_sandbox

Prevents users from temporarily changing disk sandbox settings by using the sa_server_option system procedure or the sa_db_option system procedure. The manage_disk_sandbox feature cannot be turned off for all databases or users. It can only be turned off for individual connections by using the sp_use_secure_feature_key system procedure.

server_security

This feature set prevents users from accessing features that can temporarily bypass security settings. By default, the following features, except for trace_system_event, are secured.

disk_sandbox

Prevents users from performing read-write file operations on the database outside the directory where the main database file is located.

trace_system_event

Prevents users from creating user-defined trace events.

database_isolation Allows database isolation to be temporarily turned off for the current connection.

all

This feature set prevents users from accessing the following groups:

client

This feature set prevents users from accessing all features that allow access to client-related input and output. This feature controls access to the client computing environment. This set consists of the following features:

read_client_file

Prevents the use of statements that can cause a client file to be read. For example, the READ_CLIENT_FILE function and the LOAD TABLE statement.

write_client_file

Prevents the use of all statements that can cause a client file to be written to. For example, the UNLOAD statement and the WRITE_CLIENT_FILE function.

remote

This feature set prevents users from accessing all features that allow remote access or communication with remote processes. This set consists of the following features:

remote_data_access

Prevents the use of any remote data access services, such as proxy tables.

send_email

Prevents the use of email system procedures, such as xp_sendmail.

send_udp

Prevents the ability to send UDP packets to a specified address by using the sa_send_udp system procedure.

web_service_client

Prevents the use of web service client stored procedure calls (stored procedures that issue HTTP requests).

local

This feature set prevents users from accessing all local-related features. This feature controls access to the server computing environment. This set consists of the local_call, local_db, local_io, and local_log feature subsets.

local_call

This feature set prevents users from accessing all features that can execute code that is not directly part of the database server and is not controlled by the database server. This set consists of the following features:

cmdshell

Prevents the use of the xp_cmdshell procedure.

external_procedure

Prevents the use of user-defined external stored procedures.

external_library_full_text

Prevents the use of a user-defined external term breaker library.

java

Prevents the use of Java-related features, such as Java procedures.

local_db

This feature set prevents users from accessing all features related to database files. This set consists of the following features:

backup

Prevents the use of the BACKUP DATABASE statement, and with it, the ability to run server-side backups. You can still perform client-side backups by using the dbbackup utility.

restore

Prevents the use of the RESTORE DATABASE statement.

database

Prevents the use of the CREATE DATABASE, ALTER DATABASE, and DROP DATABASE statements.

It also prevents the use of the CREATE ENCRYPTED FILE, CREATE DECRYPTED FILE, CREATE ENCRYPTED DATABASE, and CREATE DECRYPTED DATABASE statements.

dbspace

Prevents the use of the CREATE DBSPACE, ALTER DBSPACE, and DROP DBSPACE statements.

local_env

This feature set prevents users from accessing all features related to environment variables. This set consists of the following features:

getenv

Prevents users from reading the value of any environment variable.

local_io

This feature set prevents users from accessing all features that allow direct access to files and their contents. This set consists of the following features:

create_trace_file

Prevents the use of statements that create an event tracing target.

read_file

Prevents the use of statements that can cause a local file to be read. For example, the xp_read_file system procedure, the LOAD TABLE statement, and the use of OPENSTRING(FILE...).

write_file

Prevents the use of all statements that can cause a local file to be written to. For example, the UNLOAD statement and the xp_write_file system procedure.

delete_file

Prevents the use of all statements that can cause a local file to be deleted. For example, securing this feature causes the dbbackup utility to fail if the -x or -xo options are specified.

directory

Prevents the use of directory class proxy tables. This feature is disabled when remote_data_access is disabled.

file_directory_functions

This feature set prevents users from accessing all of the following individual features:

sp_list_directory

Prevents the use of the sp_list_directory system procedure.

sp_create_directory

Prevents the use of the sp_create_directory system procedure.

sp_copy_directory

Prevents the use of the sp_copy_directory system procedure.

sp_move_directory

Prevents the use of the sp_move_directory system procedure.

sp_delete_directory

Prevents the use of the sp_delete_directory system procedure.

sp_copy_file

Prevents the use of the sp_copy_file system procedure.

sp_move_file

Prevents the use of the sp_move_file system procedure.

sp_delete_file

Prevents the use of the sp_delete_file system procedure.

sp_disk_info

Prevents the use of the sp_disk_info system procedure.

local_log

Prevents users from accessing all logging features that result in creating or writing data directly to a file on disk. This set consists of the following features:

request_log

Prevents the ability to change the request log file name and also prevents the ability to increase the limits of the request log file size or number of files. You can specify the request log file and limits on this file in the command to start the database server; however, they cannot be changed once the database server is started. When request log features are disabled, you can still turn request logging on and off and reduce the maximum file size and number of request logging files.

console_log

Prevents the ability to change the database server message log file name using the ConsoleLogFile option of the sa_server_option system procedure. Securing this feature also prevents the ability to increase the maximum size of the database server message log file using the ConsoleLogMaxSize option of the sa_server_option system procedure. You can specify a server log file and its size when starting the database server.

webclient_log

Prevents the ability to change the web service client log file name using the WebClientLogFile option of the sa_server_option system procedure. You can specify a web service client log file when starting the database server.

Default

none

Applies to

All operating systems.

Remarks

This option allows the owner of the database server to control whether users have access to features for databases running on the database server. The -sk database server option allows the owner of the database server to create the SYSTEM secured feature key that permits users access to features secured by the -sf database server option.

If you start a database server without specifying the -sk database sever option, the features specified by the -sf database sever option and some default features are secured and there is no way to change which features are secured while the database server is running. You cannot create the SYSTEM secure feature key later using a system stored procedure. You must shut down the database server and specify the -sk database sever option when you restart it.

The `feature-list` is a comma-separated list of feature names or feature sets to secure for the database server. Securing a feature makes it inaccessible to all database users other than administrators. Specifying a feature set secures all the features included in the set. To secure one or more, but not all, of the features in the feature set, specify the individual feature name.

Use `feature-name` to indicate that the feature should be secured (made inaccessible), and `-feature-name` or `feature-name-` to indicate that the feature should be unsecured (accessible to all database users). For example, the following command indicates that only dbspace features are accessible to all users:

> **i** Note
>
> Sub-features of feature sets that are secured by default, cannot be unsecured from the command line. In other words the following command will not work:
>
> ```
> -sf manage_security,-manage_keys
> ```
>
> To use sub-features of feature sets that are secured by default, call the sp_use_secure_feature_key system procedure specifying the SYSTEM secure feature key which includes MANAGE_KEYS and the password specified by the -sk option :
>
> ```
> CALL sp_use_secure_feature_key( 'system' , 'letmeinweyou' );
> ```

Example

The following command starts a database server named secure_server with all local data access features secured, except for the dbspace feature.

```
dbsrv17 -n secure_server -sf all,-dbspace
```

The following command starts a database server named secure_server with all remote data access features secured, except for the web_service_client feature. The key specified by the -sk option can be used later with the sp_use_secure_feature_key system procedure to make these features accessible to all users on the current connection.

```
dbsrv17 -n secure_server -sf remote,-web_service_client -sk j978kls12
```

If a user connected to a database running on the secure_server database server uses the sp_use_secure_feature_key system procedure with the authorization_key parameter set to the same value as that specified by -sk, that connection has access to the remote data access features:

```
CALL sp_use_secure_feature_key ( 'MyKey' , 'j978kls12' );
```

The following command secures all features, with the exception of local database features:

```
dbsrv17 -n secure_server -sf all,-local_db
```

Related Information

Secured Features [page 1688]
Event Tracing [page 1012]
Access to Data on Client Computers
External Call Interface
Java in the Database
Request Logging [page 1502]
SAP SQL Anywhere Hardware Requirements
Creating Secured Feature Keys [page 1690]
Creating Secured Feature Keys [page 1690]
-sk Database Server Option [page 500]
-sbx Database Server Option [page 491]
-sbx Database Option [page 559]
xp_cmdshell System Procedure
BACKUP DATABASE Statement
RESTORE DATABASE Statement
xp_getenv System Procedure
-zoc Database Server Option [page 537]
sa_server_option System Procedure
sp_create_secure_feature_key System Procedure
sp_alter_secure_feature_key System Procedure
sp_list_secure_feature_keys System Procedure
sp_drop_secure_feature_key System Procedure
sp_use_secure_feature_key System Procedure

1.4.1.83 -sjvm Database Server Option

Specifies that the database server uses one Java VM for all databases running on the database server.

Syntax

```
dbsrv17 -sjvm { ON | OFF }
```

Default

OFF

Applies to

All operating systems except Linux ARM.

Remarks

This option specifies whether or not the database server uses one Java VM for all databases on the database server. You can also set the database server to use one Java VM for all databases by calling the sa_server_option system procedure and setting the UseSingleJVMInstance database server option to ON.

Example

The following command starts a database server named myserver1, that uses one Java VM for all databases running on the database server.

```
dbsrv17 -n myserver1 -sjvm ON
```

Related Information

sa_server_option System Procedure

1.4.1.84 -sk Database Server Option

Creates the SYSTEM secured feature key and sets the authorization code for it. This permits access to features that are secured for the database server.

Syntax

```
dbsrv17 -sk auth_code ...
```

Applies to

All operating systems.

Remarks

When you secure features for a database server by using the -sf option, you can also include the -sk option, which creates the SYSTEM secured feature key and sets the authorization code for it. This authorization code can be used with the sp_use_secure_feature_key system procedure to allow access to secured features for a connection. That connection can also use the sa_server_option system procedure to modify the features or feature sets that are secured for all databases running on the database server.

The authorization code must be a non-empty string of at least six characters, and it cannot contain double quotes, control characters (any character less than 0x20), or backslashes..

If the authorization code specified with the sp_use_secure_feature_key system procedure does not match the value specified by -sk, no error is given and the features specified by -sf remain secured for the connection.

If you specify -sk without -sf, only default features like those contained in the manage_security feature set are secured, but you can use the SYSTEM secured feature key while the database server is running to gain access to and change the secured feature settings.

Example

The following command starts a database server named secure_server with the backup feature and some default features secured. The authorization code specified by the -sk option is used later to access these features for a specific connection.

```
dbsrv17 -n secure_server -sf backup -sk j978kls12
```

You use the sp_use_secure_feature_key system procedure, specifying SYSTEM for the key name and the authorization code that was specified with the -sk option. This allows you to perform backups, change the features that are secured on the secure_server database server, and create other keys.

```
CALL sp_use_secure_feature_key( 'SYSTEM' , 'j978kls12' );
```

You can secure all features for databases running on secure_server by executing the following statement:

```
CALL sa_server_option( 'SecureFeatures', 'all' );
```

You can create other keys to permit access to selected features by other users (provided you disclose the key name and authorization code).

```
CALL sp_create_secure_feature_key ( 'client_access' , 'client_auth_code' ,
'client' );
```

Related Information

Creating Secured Feature Keys [page 1690]
Creating Secured Feature Keys [page 1690]
-sf Database Server Option [page 493]
sa_server_option System Procedure
sp_use_secure_feature_key System Procedure

1.4.1.85 -su Database Server Option

Sets the user ID and password for the utility database (utility_db), or disables connections to the utility database.

Syntax

```
dbsrv17 -su { user-ID,password | password | NONE } ...
```

Applies to

All operating systems.

Remarks

This option specifies the initial password, and optionally the user ID, for the utility database. The minimum length of the password is always 6, is case sensitive, and cannot contain a comma. Specify *NONE* instead of a user ID and/or password to disable all connections to the utility database. If you do not specify a `user-ID`, then the database server uses the default value DBA.

- If you are using a personal database server and do not specify the -su option, then the database server allows connections to the utility database with the user ID DBA and any password. Anyone who can connect to the personal database server can access the file system directly so no attempt is made to screen users based on passwords.
- If you are using a network database server and do not specify the -su database server option, then connections to the utility database are not allowed.
- Allowing connections to the utility database for a network database server is useful for cases when the database server is running, but it is not possible to connect to the database.
  For example, in a mirroring system, you can connect to the utility database to shut down the database server, or force a mirror server to become the primary server if necessary.

You can execute a CREATE USER `user-ID` IDENTIFIED BY `new-password` statement while connected to utility_db to change the password for the user `user-ID` of the utility database. The REVOKE CONNECT FROM

`user-ID` statement can be used to disable connections to the utility_db database. Not all SQL statements are supported for the utility database.

To avoid having the utility database password in clear text on the command line, you can use the dbfhide utility to encrypt a file containing `password` and then reference the encrypted file on the command line.

Example

The following command disables all connections to the utility database:

```
dbsrv17 -su none c:\inventory.db
```

In the following example, the file named `util_db_uid_pwd.cfg` that contains the utility database user ID and password is encrypted by using dbfhide and renamed `util_db_uid_pwd_hide.cfg`:

```
dbfhide util_db_uid_pwd.cfg util_db_uid_pwd_hide.cfg
```

The `util_db_pwd_hide.cfg` file can then be used to specify the utility database user ID and password:

```
dbsrv17 -su @util_db_uid_pwd_hide.cfg -n my_server c:\inventory.db
```

Related Information

The Utility Database (utility_db) [page 309]
Connecting to the Utility Database (Connect Window) [page 311]
Stopping a Database Server in a Mirroring System (dbstop Utility) [page 1781]
Troubleshooting: The Primary Server Cannot be Restarted [page 1792]
File Hiding Utility (dbfhide) [page 1160]
CREATE USER Statement

1.4.1.86 -tdsl Database Server Option

Sets the TDS login mode.

Syntax

```
dbsrv17 -tdsl { ALL | RSA | RSANONCE } ...
```

Default

ALL

Applies to

All operating systems.

Remarks

This option restricts the type of TDS login requests that the database server supports.

Mode type	Description
ALL	Encrypted and unencrypted passwords are allowed (including those with a nonce) in TDS login requests:
RSA	Only RSA encrypted passwords (including those with a nonce) are allowed in TDS login requests:
RSANONCE	Only RSA encrypted passwords with a nonce are allowed in TDS login requests:

When you make login requests from a TDS application that supports RSA without a nonce, the database server generates a new set of encryption key for the login requests. Generating new encryption keys can be time-consuming.

When you make login requests from a TDS application that supports RSA with a nonce, the database server reuses a set of RSA encryption keys. These encryption keys are re-generated every 24 hours. By reusing the RSA encryption keys, performance can improve while protecting the database server from replay attacks.

Both jConnect and Open Client support RSA login requests with and without a nonce.

Related Information

SAP Open Client Support

1.4.1.87 -ti Database Server Option

Disconnects inactive connections.

Syntax

```
dbsrv17 -ti minutes ...
```

Default

240 (4 hours)

Applies to

All operating systems.

Remarks

Disconnects connections that haven't submitted a request for the specified number of `minutes`. The maximum value is 32767. A client computer in the middle of a database transaction holds locks until the transaction is ended or the connection is disconnected. The -ti option is provided to disconnect inactive connections, freeing their locks.

Setting the value to zero disables checking of inactive connections, so that no connections are disconnected. If the IdleTimeout (IDLE) connection parameter is not used, then the idle timeout value for TCP/IP connections is controlled by the -ti database server option. If both the -ti database server option and the IdleTimeout (IDLE) connection parameter are specified, then the idle timeout value is controlled by the connection parameter.

Idle timeout is not supported for TDS clients, including jConnect clients.

Related Information

-tl Database Server Option [page 506]
sa_server_option System Procedure
Troubleshooting Network Communications [page 1028]
IdleTimeout (IDLE) Connection Parameter [page 89]

1.4.1.88 -tl Database Server Option

Sets the period at which to send liveness packets.

Syntax

```
dbsrv17 -tl seconds ...
```

Applies to

All operating systems. TCP/IP only.

Remarks

A liveness packet is sent periodically across a client/server TCP/IP communications protocol to confirm that a connection is intact. If the server runs for a LivenessTimeout period (default 2 minutes) without detecting a liveness packet on a connection, the communication is severed, and the server drops the connection associated with that client. UNIX and Linux non-threaded clients and TDS connections do not do liveness checking.

The -tl option on the server sets the LivenessTimeout value for all clients that do not specify a liveness period.

Liveness packets are sent when a connection hasn't sent any packets for between one third and two thirds of the LivenessTimeout value.

When there are more than 200 connections, the server automatically calculates a higher LivenessTimeout value based on the stated LivenessTimeout value, so the server can handle a large number of connections more efficiently. Liveness packets are sent between one third and two thirds of the LivenessTimeout on each idle connection. Large numbers of liveness packets aren't sent at the same time. Liveness packets that take a long time to send could be sent after two thirds of the LivenessTimeout. A warning appears in the database server message log if the liveness sends take a long time. If this warning occurs, consider increasing the LivenessTimeout value.

Although it isn't generally recommended, you can disable liveness by specifying the following:

```
dbsrv17 -tl 0 -n my_server
```

Rather than disabling the LivenessTimeout option, consider increasing the value to one hour as follows:

```
dbsrv17 -tl 3600 -n my_server
```

Related Information

-ti Database Server Option [page 505]

sa_server_option System Procedure
Troubleshooting Network Communications [page 1028]

1.4.1.89 -tmf Database Server Option

Forces transaction manager recovery for distributed transactions.

Syntax

```
dbsrv17 -tmf ...
```

Applies to

Windows.

Remarks

Used during recovery of distributed transactions when the distributed transaction coordinator isn't available. If DTC cannot be located, the outstanding operations are rolled back and recovery continues. It can also be used when starting a database with distributed transactions in the transaction log on a platform where the DTC isn't available.

Caution

If you use this option, distributed transactions are not recovered properly. It is not intended for routine use.

Related Information

Recovery from Distributed Transactions
-tmt Database Server Option [page 507]

1.4.1.90 -tmt Database Server Option

Sets a re-enlistment timeout for participation in distributed transactions.

Syntax

```
dbsrv17 -tmt milliseconds ...
```

Applies to

Windows.

Remarks

Used during recovery of distributed transactions. The value specifies how long the database server should wait to be reenlisted. By default there is no timeout (the database server waits indefinitely).

Related Information

Recovery from Distributed Transactions
-tmf Database Server Option [page 507]

1.4.1.91 -tq Database Server Option

Shuts down the server at a specified time.

Syntax

```
dbsrv17 -tq { datetime | time } ...
```

Applies to

All operating systems.

Remarks

This option is useful for setting up automatic off-line backup procedures.

The format for the time is in `hh:mm` (24 hour clock), and can be preceded by an optional date. If a date is specified, the date and time must be enclosed in double quotes and be in the format `YYYY/MM/DD HH:MM`.

Related Information

Database Backup and Recovery [page 939]
sa_server_option System Procedure

1.4.1.92 -ts Database Server Option

Sets up a database server tracing session.

Syntax

```
dbsrv17 -ts session-name ( session-option=option-value [ ;... ] )
```

Session option	Option value
events	Comma separated list of system trace events. Call the sp_trace_events system procedure to obtain a list of system-defined trace events.
targets	`target-type` (`target-option=value` [;...]) where `target-type` can only be *file*.

The target file can have the following options:

Target option	Option value
filename_prefix	An ETD file name prefix with or without a path. All ETD files have the extension `.etd`. This parameter is required.
max_size	The maximum size of the file in bytes. The default is 0, which means there is no limit on the file size and it grows as long as disk space is available. Once the specified size is reached, a new file is started.
num_files	The number of files where event tracing information is written, and it is used only if max_size is set. If all the files reach the maximum specified size, the database server starts overwriting the oldest file.
flush_on_write	A value that controls whether disk buffers are flushed for each event that is logged. The values yes, true, no, and false are accepted. The default is false. When this parameter is turned on, the performance of the database server may be reduced if many trace events are being logged.
compressed	A value that controls compression of the ETD file to conserve disk space. The values on and off are accepted. The default is off.

Applies to

All operating systems.

Remarks

Server trace event sessions can be used to capture trace events related to system behavior or for a particular user. Server trace event sessions are stored in memory and are dropped when the database server stops, if they have not been dropped explicitly.

Related Information

Event Tracing [page 1012]
System Events [page 1001]
CREATE TEMPORARY TRACE EVENT SESSION Statement
CREATE TEMPORARY TRACE EVENT Statement
ALTER TRACE EVENT SESSION Statement
DROP TRACE EVENT Statement
DROP TRACE EVENT SESSION Statement
NOTIFY TRACE EVENT Statement
sp_trace_events System Procedure
sp_trace_event_fields System Procedure
sp_trace_event_sessions System Procedure
sp_trace_event_session_events System Procedure
sp_trace_event_session_targets System Procedure
sp_trace_event_session_target_options System Procedure
Event Trace Data (ETD) File Management Utility (dbmanageetd) [page 1157]
-sf Database Server Option [page 493]

1.4.1.93 -u Database Server Option

Opens files using the operating system disk cache.

Syntax

```
dbsrv17 -u ...
```

Applies to

All operating systems.

Remarks

Files are opened using the operating system disk cache in addition to the database cache.

While the operating system disk cache may improve performance, better performance is often obtained by using only the database cache.

If the server is running on a dedicated computer, you shouldn't use the -u option, as the database cache itself is generally more efficient. You may want to use the -u option if the server is running on a computer with several other applications (so that a large database cache may interfere with other applications) and yet IO-intensive tasks are run intermittently on the server (so that a large cache will improve performance).

1.4.1.94 -ua Database Server Option

Turns off use of asynchronous I/O.

Syntax

```
dbsrv17 -ua ...
```

Applies to

Linux.

Remarks

By default, the database server uses asynchronous I/O on Linux when possible. To use asynchronous I/O, the following conditions must be met:

1. The library `libaio.so` can be loaded at run time.
2. The kernel has asynchronous I/O support.

To turn off the use of asynchronous I/O, specify the -ua database server option.

1.4.1.95 -uc Database Server Option

Starts the database server in shell mode.

Syntax

```
dbsrv17 -uc ...
```

Applies to

Unix.

Remarks

Starts the database server in shell mode; this is the default. You should only specify one of -uc, -ui, -um, -uq, or -ux. When you specify -uc, this starts the database server in the same manner as previous releases of the software.

You can also start the database server as a daemon using the -ud database server option.

Related Information

-ui Database Server Option [page 516]
-um Database Server Option [page 517]
-ux Database Server Option [page 519]
-ud Database Server Option [page 512]

1.4.1.96 -ud Database Server Option

Runs the database server as a daemon.

Syntax

```
dbsrv17 -ud ...
```

Applies to

Unix.

Remarks

Using this option lets you run the server so that it continues running after the current user session ends.

When you start the daemon directly using the -ud option, the database server create the daemon process and return immediately (exiting and allowing the next command to be executed) before the daemon initializes itself or attempts to open any of the databases specified in the command.

One advantage of using dbspawn instead of the -ud option is that the dbspawn process does not shut down until it has confirmed that the daemon has started and is ready to accept requests. If for any reason the daemon fails to start, the exit code for dbspawn is non-zero.

When you start the database server as a daemon, its permissions are controlled by the current user's unmask setting. Set the unmask value before starting the database server to ensure that the database server has the appropriate permissions.

Do not specify the -um option with the -ud option.

Related Information

How to Run the Database Server as a Service or Daemon [page 354]
General Security Tips [page 1680]
Start Server in Background Utility (dbspawn) [page 1217]
Software Component Exit Codes

1.4.1.97 -uf Database Server Option

Specifies the action that the database server takes when a fatal error or assertion failure occurs on a database server.

Syntax

```
dbsrv17 -uf action ...
```

Default

default (for UNIX and Linux)

defunct (for Windows)

Allowed Values

abort

The database server is shut down and a core file is generated.

default

The database server behaves in the same manner as abort, except when a device-full fatal error occurs. In this case, it behaves in the same manner as defunct. This action prevents the system from trying to write a core file on a full device.

defunct

The database server continues running and does not call abort. Any new connection attempts that are made to the database server receive the original error.

Applies to

All operating systems.

Related Information

Database Server Logging [page 336]
Troubleshooting: Reporting an Error [page 1021]
-oe Database Server Option [page 472]
Support Utility (dbsupport) [page 1221]
-ufd Database Server Option [page 514]

1.4.1.98 -ufd Database Server Option

Specifies the action that the database server takes when a fatal error or assertion failure occurs on a database.

Syntax

```
dbsrv17 -ufd action ...
```

Default

abort

Allowed Values

abort

The affected database is stopped. The status of the database server and other databases remains unchanged.

escalate

The database assertion failure becomes a database server assertion failure and is handled according to the setting for the -uf database server option.

restart

This value is deprecated; use restart_escalate instead.

restart_abort

The database server automatically restarts the database. A restart is considered unsuccessful if another database assertion failure occurs within 60 seconds of the restart. If the second consecutive restart is unsuccessful, the affected database is stopped. The status of the database server and other databases remains unchanged.

restart_escalate

The database server automatically restarts the database. A restart is considered unsuccessful if another database assertion failure occurs within 60 seconds of the restart. If the second consecutive restart is unsuccessful, then the database assertion failure causing the unsuccessful restart becomes a server assertion failure and is handled according to the setting for the -uf database server option.

Applies to

All operating systems.

Remarks

The -ufd database server option must precede any database file specification.

Related Information

Database Server Logging [page 336]

Troubleshooting: Reporting an Error [page 1021]
-oe Database Server Option [page 472]
Support Utility (dbsupport) [page 1221]
-uf Database Server Option [page 513]

1.4.1.99 -ui Database Server Option

Opens the *Server Startup Options* window, displays the database server messages window, and starts the database server whether the X window server starts or not.

Syntax

```
dbsrv17 -ui ...
```

Applies to

Linux with X window server support.

Remarks

The -ui option allows you to use the *Server Startup Options* window to specify server options when starting the database server, and to display the database server messages window once the database server has started.

When the -ui database server option is the only option specified, the *Server Startup Options* window appears where you can enter options for starting the database server.

The database server attempts to find a usable display when the -ui database server option is specified. If it cannot find one, for example because the DISPLAY environment variable isn't set or because X window server isn't running, then the database server starts in shell mode. If you do not want the database server to start when it cannot locate a usable display, specify the -ux database server option rather than -ui. You should only specify one of -uc, -ui, -um, -uq, or -ux.

You can also start the database server as a daemon using the -ud database server option.

Related Information

-uc Database Server Option [page 512]
-um Database Server Option [page 517]
-ux Database Server Option [page 519]
-ud Database Server Option [page 512]

1.4.1.100 -um Database Server Option

Displays database server messages in a new window within `DBLauncher.app`.

Syntax

```
dbsrv17 -um ...
```

Applies to

macOS.

Remarks

The -um option allows you to connect to the `DBLauncher.app` instance, if it is running, and displays messages in a new window within `DBLauncher.app`. The -um option must be used with the other options required to start the database server. Server messages appear in this window instead of in the shell. Closing this window shuts down the database server. If a connection to the `DBLauncher.app` instance cannot be established, the database server does not start.

For the database server to connect to a `DBLauncher.app` instance, both must be running in the same macOS security context. For example, a database server started from an SSH session cannot find a `DBLauncher.app` instance that was started by Launch Services. You should only specify one of -uc, -ui, -um, -uq, or -ux.

You can also start the database server as a daemon using the -ud database server option.

Do not specify the -um option with the -ud option.

Related Information

-uc Database Server Option [page 512]
-ui Database Server Option [page 516]
-ud Database Server Option [page 512]

1.4.1.101 -uq Database Server Option

Starts the database server in shell mode but suppresses output.

Syntax

```
dbsrv17 -uq ...
```

Applies to

UNIX and Linux.

Remarks

Starts the database server in shell mode but suppresses output. The database server does not stop if the Q key is pressed. Specify only one of -uc, -ui, -um, -uq, or -ux.

1.4.1.102 -ut Database Server Option

Touches temporary files.

Syntax

```
dbsrv17 -ut minutes ...
```

Default

The default is 30 minutes.

Applies to

UNIX and Linux.

Remarks

This option causes the server to touch temporary files at specified intervals.

1.4.1.103 -ux Database Server Option

Opens the *Server Startup Options* window or displays the database server messages window on Linux (use the X window server).

Syntax

```
dbsrv17 -ux ...
```

Applies to

Linux with X window server support.

Remarks

The -ux option allows you to do two things when starting the database server: use the *Server Startup Options* window to specify server options when starting the database server and display the database server messages window once the server has started.

When the -ux database server option is the only option specified, the *Server Startup Options* window appears where you can enter options for starting the database server.

The server must be able to find a usable display when -ux is specified. If it cannot find one, for example because the DISPLAY environment variable isn't set or because X window server isn't running, then the database server fails to start. If you want the database server to start, even if it cannot find a usable display, use the -ui option instead of -ux.

If you specify other server options in addition to -ux, then the database server messages window appears once the database server is started. You should only specify one of -uc, -ui, -um, -uq, or -ux.

You can start the database server as a daemon using the -ud database server option.

Example

The following command displays the *Server Startup Options* window where you can enter options for starting the database server:

```
dbsrv17 -ux
```

The following command starts the database server and displays the database server messages window:

```
dbsrv17 -ux sample.db
```

Related Information

-uc Database Server Option [page 512]
-ui Database Server Option [page 516]
-qn Database Server Option [page 484]
-ud Database Server Option [page 512]

1.4.1.104 -v Database Server Option

Displays the software version.

Syntax

```
dbsrv17 -v ...
```

Applies to

All operating systems.

Remarks

Supplies the database server version in a window, and then stops.

1.4.1.105 -vss Database Server Option

Enables and disables the use of the Volume Shadow Copy Service (VSS).

Syntax

```
dbsrv17 -vss{ + | - } ...
```

Applies to

32-bit and 64-bit editions of Microsoft Windows 2003 and later operating systems.

Allowed Values

-vss+ Enable the use of the Volume Shadow Copy Service (VSS).
-vss- Disable the use of the Volume Shadow Copy Service (VSS).

Remarks

By default, if the SQL Anywhere VSS writer (`dbvss17.exe`) is running, then all databases can use the VSS service for backups. You can use VSS without the SQL Anywhere VSS writer to back up databases. However, you might need to use the full SQL Anywhere recovery procedures to restore those databases. To prevent a database server from participating in the VSS service, include *-vss-* when starting the database server.

Example

The following command starts the `mydatabase.db` database and instructs the database server not to participate in VSS operations even if the (`dbvss17.exe`) writer is running:

```
dbsrv17 -vss- mydatabase.db
```

Related Information

SQL Anywhere Volume Shadow Copy Service (VSS) [page 945]
Recovering from Media Failure on the Database File [page 984]
Service Utility (dbsvc) for Windows [page 1212]

1.4.1.106 -wc Database Server Option

Controls whether checksums are enabled on write operations for all databases on this database server, if the databases do not have checksums enabled by default.

Syntax

```
dbsrv17 -wc[ + | - ] ...
```

Applies to

All operating systems.

Allowed Values

-wc+ Enable checksums for all databases.

-wc Disable checksums for all databases that do not have checksums automatically enabled.

-wc- Disable checksums for all databases that do not have checksums automatically enabled.

Remarks

The difference between the write checksums (enabled with the -wc option) and global checksums (creating a database with checksums enabled) is that with -wc database pages are checksummed only when they are written out to disk. Pages that are read from disk are only verified if a checksum value was calculated before the pages were written. If a database has checksums enabled, checksums are calculated for all pages when they are written and checksums are verified for all pages when they are read.

If the database server detects that the database is running on a removable storage device, such as a network share or USB device, then the database server automatically enables global checksums for all database pages.

By default, databases created with version 10 and 11 of SQL Anywhere do not have global checksums enabled. If you start a database created with SQL Anywhere 11 on a version 12 or later database server, then by default the database server creates write checksums for pages when they are written to disk (-wc+). Version 12 and later databases have global checksums enabled by default, so the database server defaults to -wc- for these databases because by default all database pages have checksums. You can use either the -wc option or the START DATABASE statement to change the database server's checksum behavior if you do not want to use the default checksum settings.

You can check whether a database was created with global checksums enabled by executing the following statement:

```
SELECT DB_PROPERTY ( 'Checksum' );
```

You can check whether write checksums are enabled for write I/O operations only by executing the following statement:

```
SELECT DB_PROPERTY ( 'WriteChecksum' );
```

Related Information

Corruption Detection Using Checksums [page 993]
START DATABASE Statement

1.4.1.107 -x Database Server Option

Specifies server-side network communications protocols.

Syntax

```
dbsrv17 -x { ALL | NONE | TCPIP [ ( parm=value;...) } ...
```

Allowed Values

Regardless of which settings you choose for the -x option, the database server always accepts shared memory connections.

By default, the personal database server starts only the shared memory protocol.

By default, the network database server starts the shared memory and TCP/IP protocols.

You can specify the following values for the -x option:

ALL

Listen for connection attempts by the client using shared memory and TCP/IP protocols.

NONE

Listen for connection attempts by the client using only the shared memory protocol.

TCPIP (TCP)

Listen for connection attempts by the client using the shared memory and TCP/IP protocols.

The TCP/IP protocol is supported by the network server, and by the personal database server for same-computer communications.

Applies to

All operating systems.

Remarks

Use the -x option to specify which communications protocols you want to use to listen for client connections.

For the TCP/IP protocol, additional parameters may be provided, in the following format:

```
-x tcpip(parm1=value1;parm2=value2;...)
```

For UNIX and Linux, quotation marks are required if more than one parameter is supplied or certain punctuation characters are used, such as parentheses and semicolons. For example:

```
-x "tcpip(parm1=value1;parm2=value2;...)"
```

When the database server listens for TCP/IP connections, by default it listens to all network adapters on port 2638.

Example

Allow shared memory and TCP/IP connections on the network server:

```
dbsrv17 -n server_name "%SQLANYSAMP17%\demo.db"
```

Use the -x option to specify protocol options to tune the behavior of TCP/IP. Allow the network server to use two specific network cards:

```
dbsrv17 -x "tcpip(MyIP=192.75.209.12,192.75.209.32)" "%SQLANYSAMP17%\demo.db"
```

Allow shared memory and TCP/IP connections on the personal server:

```
dbeng17 -x tcpip "%SQLANYSAMP17%\demo.db"
```

Related Information

General Security Tips [page 1680]
Network Protocol Options [page 187]
How to Start the Database Server [page 324]
TCP/IP Protocol [page 176]
-xd Database Server Option [page 526]
-xp Database Option [page 563]

-xs Database Server Option [page 529]
CommLinks (LINKS) Connection Parameter [page 59]

1.4.1.108 -xa Database Server Option

Specifies a comma-separated list of database names and authentication strings for an arbiter server.

Syntax

```
dbsrv17 -xa [ AUTH=auth-string [, auth-string... ]; ] DBN={ * | database-name
[, database-name... ] } ...
```

On Unix, quotation marks are required when more than one parameter is supplied:

Applies to

All operating systems.

Network database server (dbsrv17) only.

Remarks

This option is only specified when starting the arbiter server in a database mirroring system.

AUTH parameter

Omitting the authentication string means that there is no validation of the authentication string provided by a partner server.

If only one authentication string is specified, all databases must use that authentication string.

If more than one authentication string is provided in the list of authentication strings, then the list of database names must contain the same number of entries. The authentication string is verified with the corresponding database in the DBN list.

The authentication string must match the SET MIRROR OPTION `authentication_string` value for all databases that use this server as an arbiter.

DBN parameter

This parameter lists the databases that can use this server as an arbiter. To allow any database to use the server as an arbiter, specify **`DBN=*`**.

Example

The following command starts the arbiter database server and specifies that any database can use the server as an arbiter:

```
dbsrv17
-n arbiter
-su passwd
-x tcpip(port=6870)
-xf c:\arbiter\arbiterstate.txt
-xa "AUTH=abc;DBN=*"
```

The following command starts the arbiter database server and specifies the two databases that can use the server as an arbiter:

```
dbsrv17
-n arbiter
-su passwd
-x tcpip(port=6870)
-xf c:\arbiter\arbiterstate.txt
-xa "AUTH=abc,xyz;DBN=database1,database2"
```

In this example, database1 uses authentication string abc and database2 uses authentication string xyz.

Related Information

Tutorial: Creating a Database Mirroring System [page 1800]
Tutorial: Creating a Database Mirroring System with Multiple Databases That Share an Arbiter Server [page 1808]
Adding a Mirrored Database to a Running Mirroring System [page 1784]
CREATE MIRROR SERVER Statement
SET MIRROR OPTION Statement
DatabaseName (DBN) Connection Parameter [page 71]
-xf Database Server Option [page 527]
-xp Database Option [page 563]

1.4.1.109 -xd Database Server Option

Prevents the database server from becoming the default database server.

Syntax

```
dbsrv17 -xd ...
```

Applies to

All operating systems.

Remarks

When a database server starts, it attempts to become the default database server on that computer. The first database server to start when there is no default server becomes the default database server. Shared memory connection attempts on that computer that do not explicitly specify a database server name connect to the default server.

Specifying this option prevents the database server from becoming the default database server. If this option is specified, clients that do not specify a database server name cannot find the database server over shared memory. The -xd option also prevents the database server from using the default TCP port. If a TCP port is not specified, the database server uses a port other than port 2638.

Related Information

-n Database Server Option [page 467]
StartLine (START) Connection Parameter [page 112]
-x Database Server Option [page 523]
ServerName (Server) Connection Parameter [page 110]
Troubleshooting: How Database Servers Are Located [page 263]

1.4.1.110 -xf Database Server Option

Specifies the location of the file used for maintaining state information about your database mirroring system.

This option is only used in the command to start the arbiter server in a database mirroring system.

Syntax

```
dbsrv17 -xf state-file ...
```

Applies to

All operating systems.

Network database server (dbsrv17) only.

Remarks

Use the CREATE MIRROR SERVER to define the location of the state information file for the partner servers.

The -xf option specifies the location of the file used for maintaining state information about the mirroring system. This option is required for database mirroring.

Example

The following command starts a database server named myarbiter, that uses the state information file `c:\arbiter\arbiter.state`.

```
dbsrv17 -n myarbiter -su passwd -x "TCPIP(PORT=6870;DOBROAD=no)" -xf "c:\arbiter
\arbiter.state" -xa "AUTH=abc;DBN=mirror_demo"
```

Related Information

Troubleshooting: State Information Files of the Partners and Arbiter [page 1790]
Tutorial: Creating a Database Mirroring System [page 1800]
Tutorial: Creating a Database Mirroring System with Multiple Databases That Share an Arbiter Server [page 1808]
CREATE MIRROR SERVER Statement
-xa Database Server Option [page 525]
-xp Database Option [page 563]

1.4.1.111 -xm Database Server Option

Specifies how often the database server checks for new IP addresses.

Syntax

```
dbsrv17 -xm seconds
```

Applies to

All operating systems.

Remarks

If the computer on which the database server is running is connected to a new network and the -xm option is specified, the change is detected and the database server starts listening for connections on the new network. Also, if the computer is disconnected from a network, the database server stops listening on that network.

The -xm option is set to 0 by default if the SQL Anywhere application is not running on a portable device. The default setting on portable devices is 120 seconds. To disable the -xm option, specify 0. Use the IsPortableDevice property to check if the SQL Anywhere application is running on a portable device.

Specifying this option does not affect the performance of HTTP or HTTPS listeners.

If you have specified the MyIP (ME) network protocol option, monitoring with the -xm option is disabled.

If a network interface is disconnected, all listeners associated with the network interface are shut down.

Use the sa_server_option system procedure to change the setting of this option without shutting down the database server.

Example

The following command starts a database server that checks for new IP addresses every 60 seconds.

```
dbsrv17 -xm 60
```

Related Information

sa_server_option System Procedure
MyIP (ME) Protocol Option [page 229]

1.4.1.112 -xs Database Server Option

Specifies server-side web services communications protocols.

Syntax

```
dbsrv17 -xs { NONE
| HTTP [ ( parm=value;...) ]
| HTTPS [ ( parm=value;...) ]
| ODATA [ ( odata-parm=value;...) ] } ...
```

On UNIX and Linux, quotation marks are required if more than one parameter is supplied.

Allowed Values

You can specify any of the following:

HTTP

Listen for web requests by the client using the HTTP protocol. The default port on which to listen is 80.

HTTPS

Listen for web requests by the client using the HTTPS protocol. The default port on which to listen is 443. Specify the server's certificate and password to use HTTPS. The certificate must be an RSA certificate because HTTPS uses RSA encryption.

The HTTP server supports HTTPS connections using TLS version 1.0 or later.

Specify HTTPS, or HTTPS with FIPS=Y for FIPS-certified RSA encryption. FIPS-certified HTTPS uses a separate certified library, but is compatible with HTTPS.

NONE

Do not listen for web requests. This is the default.

ODATA

Listen for web requests by the client using the OData protocol. Specifying -xs ODATA starts an OData server that runs as a sub-process of the database server. For more information about OData protocol options, see the *Network protocol options* topic.

> **i** Note
>
> Specifying -xs ODATA on platforms other than Windows or Linux (not ARM) returns an error.

parm

A network protocol option.

odata-parm

An OData protocol option.

Applies to

All operating systems.

Remarks

Use the -xs option to specify which web protocols you want to use to listen for requests.

If you do not specify the -xs option, then the database server doesn't attempt to listen for web requests.

To specify multiple protocols, specify the -xs option for each protocol or specify multiple protocols with one -xs database server option. The database server listens for web requests using all the specified protocols.

If you specify both the HTTP and OData protocols, then at least one of them must specify a non-default HTTP port number. If you specify both the HTTPS and OData protocols, then at least one of them must specify a non-default HTTPS port number.

> **i Note**
>
> To start multiple web servers at the same time, change the port for one of them since they all have the same default port.

Use the HTTPS and FIPS-validated HTTPS protocol for transport layer security.

Example

Listen for HTTP web requests on port 80:

```
dbsrv17 web.db -xs HTTP(PORT=80)
```

Listen for web requests using HTTPS:

```
dbsrv17 web.db -xs
HTTPS(FIPS=N;PORT=82;IDENTITY=ecserver.id;IDENTITY_PASSWORD=test)
```

Listen for web requests using HTTPS with the minimum TLS version to 1.2 to avoid weak TLS 1.0 and 1.1:

```
dbsrv17 web.db -xs
HTTPS(FIPS=N;PORT=82;IDENTITY=ecserver.id;IDENTITY_PASSWORD=test;MIN_TLS_VERSION=
1.2)
```

Listen on ports 80 and 8080:

```
dbsrv17 -xs HTTP(port=80),HTTP(PORT=8080)
```

or

```
dbsrv17 -xs HTTP(port=80) -xs HTTP(PORT=8080)
```

Start a database server that listens on port 8080 for OData requests:

```
dbsrv17 -xs odata(ServerPort=8080) c:\mydatabase.db
```

Run the following command to start a database server that listens on port 8443 for secure OData requests using the key store `c:\mykeys.jks` and the key store password *password*, and specifying that the database server does use the default port to listen for non-secure OData requests:

```
dbsrv17 -xs odata(SecureServerPort=8443;SSLKeyStore=c:
\mykeys.jks;SSLKeyStorePassword=password;ServerPort=)
```

Related Information

Network Protocol Options [page 187]

SQL Anywhere Web Services Encryption [page 1751]
The Database Server as an HTTP Web Server
-fips Database Server Option [page 426]
-sn Database Option [page 560]
-x Database Server Option [page 523]
-xa Database Server Option [page 525]
-xf Database Server Option [page 527]
-xp Database Option [page 563]
FIPS Protocol Option [page 205]
CREATE ODATA PRODUCER Statement
ALTER ODATA PRODUCER Statement
DROP ODATA PRODUCER Statement
sp_start_listener System Procedure
sp_stop_listener System Procedure

1.4.1.113 -z Database Server Option

Displays diagnostic communication messages, and other messages, for troubleshooting purposes.

Syntax

```
dbsrv17 -z ...
```

Applies to

All operating systems.

Remarks

This should only be used when tracking problems. The information appears in the database server messages window.

Related Information

-ze Database Server Option [page 533]
sa_server_option System Procedure

1.4.1.114 -ze Database Server Option

Displays database server environment variables in the database server messages window.

Syntax

```
dbsrv17 -ze ...
```

Applies to

All operating systems.

Remarks

When you specify the -ze option, environment variables are listed in the database server messages window on startup. You can log the contents of the database server messages window to a file by specifying the -o option when starting the database server.

Example

The following command starts a database server named myserver, and outputs the environment variables set for the server to the database server messages window and the file `server-log.txt`.

```
dbsrv17 -n myserver -ze -o server-log.txt
```

Related Information

Environment Variables [page 565]
-o Database Server Option [page 471]
-z Database Server Option [page 532]

1.4.1.115 -zl Database Server Option

Turns on capturing of the most recently prepared SQL statement for each connection to databases on the server.

Syntax

```
dbsrv17 -zl ...
```

Applies to

All operating systems.

Remarks

This feature can also be turned on using the RememberLastStatement server setting. You can obtain the most recently prepared SQL statement for a connection using the LastStatement value of the CONNECTION_PROPERTY function. The sa_conn_activity stored procedure allows you to obtain the most recently prepared SQL statement for all current connections to databases on the server.

The LastStatement value is set when a statement is prepared, and is cleared when a statement is dropped. Only one statement string is remembered for each connection.

If sa_conn_activity reports a non-empty value for a connection, it is most likely the statement that the connection is currently executing. If the statement had completed, it would likely have been dropped and the property value would have been cleared. If an application prepares multiple statements and retains their statement handles, the LastStatement value does not reflect what a connection is currently doing.

For stored procedure calls, only the outermost procedure call appears, not the statements within the procedure.

⚠ Caution

When -zl is specified or when the RememberLastStatement server setting is turned on, any user can call the sa_conn_activity system procedure or obtain the value of the LastStatement connection property to find out the most recently prepared SQL statement for any other user. The user must have either the SERVER OPERATOR, MONITOR, or DROP CONNECTION system privilege.

Related Information

sa_conn_activity System Procedure
sa_server_option System Procedure

1.4.1.116 -zn Database Server Option

Specifies the number of request log file copies to retain.

Syntax

```
dbsrv17 -zn integer ...
```

Applies to

All operating systems.

Remarks

If request logging is enabled over a long period of time, the request log file can become large. The -zn option allows you to specify the number of request log file copies to retain. It only takes effect if -zs is also specified. The -zs option allows you to create a new log file and rename the original log file when the original log file reaches a specified size.

For example, if you redirect request logging information to the file `req.out`, and specify five request log file copies using the -zn option, the server creates files in the following order: `req.out.1`, `req.out.2`, `req.out.3`, `req.out.4`, and `req.out.5`. When these files exist and the active request log fills again, the following happens:

- `req.out.1` is deleted
- the files `req.out.2` to `req.out.5` are renamed `req.out.1` to `req.out.4`
- the copy of the active log is renamed `req.out.5`

Request logging is turned on using the -zr option and redirected to a separate file using the -zo option. You can also set the number of request logs using the sa_server_option system procedure where `nn` specifies the number of request log file copies:

```
CALL sa_server_option('RequestLogNumFiles',nn);
```

Example

In the following example, request logging information is output to a request log file named `mydatabase.log`, which has a maximum size of 10 KB, and three copies of the request log are kept:

```
dbsrv17 "c:\my data\mydatabase.db" -zr all -zn 3
 -zs 10 -zo mydatabase.log
```

Related Information

Request Logging [page 1502]
-zs Database Server Option [page 540]
-zo Database Server Option [page 536]
-zr Database Server Option [page 538]
-zs Database Server Option [page 540]
sa_server_option System Procedure

1.4.1.117 -zo Database Server Option

Redirects request logging information to a file separate from the regular log file.

Syntax

```
dbsrv17 -zo filename ...
```

Applies to

All operating systems.

Remarks

Request logging is turned on using the -zr option. You can direct the output from this file to a different file that is not the regular log file by specifying the -zo option.

This option also prevents request logging from appearing in the database server messages window.

Related Information

Request Logging [page 1502]
-zn Database Server Option [page 535]
-zr Database Server Option [page 538]
-zs Database Server Option [page 540]
sa_server_option System Procedure

1.4.1.118 -zoc Database Server Option

Redirects web service client information to a file.

Syntax

```
dbsrv17 -zoc filename ...
```

Applies to

All operating systems.

Remarks

The web service client log file contains HTTP requests and transport data recorded for outbound web service client calls. The web service client log file can also be specified using the sa_server_option system procedure:

```
CALL sa_server_option( 'WebClientLogFile', 'clientinfo.txt' );
```

Logging is enabled automatically when you specify the -zoc server option. You can enable and disable logging to this file using the sa_server_option system procedure:

```
CALL sa_server_option( 'WebClientLogging', 'ON' );
```

Example

The following command starts the database server so that it listens for HTTP web requests on port 80, and logs outbound web service client information to the file `clientinfo.txt`:

```
dbsrv17 web.db -xs HTTP(PORT=80) -zoc clientinfo.txt
```

Related Information

The Database Server as an HTTP Web Server
sa_server_option System Procedure
CREATE FUNCTION Statement [Web Service]
CREATE PROCEDURE Statement [Web Service]

1.4.1.119 -zp Database Server Option

Turns on capturing of the plan most recently used by the query optimizer.

Syntax

```
dbsrv17 -zp ...
```

Applies to

All operating systems.

Remarks

Include this option if you want the database server to store the query execution plan that was used most recently by each connection. This feature can also be turned on using the RememberLastPlan server setting with the sa_server_option system procedure. You can view the text of the most recently used plan by using the LastPlanText connection property.

Related Information

sa_conn_activity System Procedure
sa_server_option System Procedure

1.4.1.120 -zr Database Server Option

Enables request logging of operations.

Syntax

```
dbsrv17 -zr { SQL | HOSTVARS | PLAN | PROCEDURES | TRIGGERS | OTHER |
BLOCKS | REPLACE | ALL | NONE } ...
```

Allowed Values

SQL

enables logging of the following:

- START DATABASE statements
- STOP DATABASE statements
- STOP SERVER statements
- Statement preparation and execution
- EXECUTE IMMEDIATE statement
- Option settings
- COMMIT statements
- ROLLBACK statements
- PREPARE TO COMMIT operations
- Connects and disconnects
- Beginnings of transactions
- DROP STATEMENT statements
- Cursor explanations
- Cursor open, close, and resume
- Errors

HOSTVARS

enables logging of host variable values. If you specify HOSTVARS, the information listed for SQL is also logged.

PLAN

enables logging of execution plans (short form). Execution plans for procedures are also recorded if logging of procedures (PROCEDURES) is enabled.

PROCEDURES

enables logging of statements executed from within procedures and user-defined functions.

TRIGGERS

enables logging of statements executed from within triggers.

OTHER

enables logging of additional request types not included by SQL, such as FETCH and PREFETCH. However, if you specify OTHER but do not specify SQL, it is the equivalent of specifying SQL+OTHER. Including OTHER can cause the request log file to grow rapidly and could negatively affect server performance.

BLOCKS

enables logging of details showing when a connection is blocked and unblocked on another connection.

REPLACE

at the start of logging, the existing request log is replaced with a new (empty) one of the same name. Otherwise, the existing request log is opened and new entries are appended to the end of the file.

ALL

logs all supported information. This setting is equivalent to specifying SQL+PLAN+HOSTVARS +PROCEDURES+TRIGGERS+OTHER+BLOCKS. This setting can cause the request log file to grow rapidly and could negatively affect server performance.

NO or NONE

turns off logging to the request log.

Applies to

All operating systems.

Remarks

This option should only be used when tracking problems. The information appears in the database server messages window or is sent to the request log. When you specify multiple values, they are separated with a , or a +.

Once the database server is started, you can change the request log settings to log more or less information using the sa_server_option system procedure.

> **Note**
>
> The -zo database server option is recommended whenever using the -zr database server option. Using the -zr database server option without using the -zo database server option can impact database server performance and in some cases increase memory usage.

You can find the current value of the RequestLogging setting using the following query:

```
SELECT PROPERTY( 'RequestLogging' );
```

Related Information

Request Logging [page 1502]
sa_server_option System Procedure
-zn Database Server Option [page 535]
-zo Database Server Option [page 536]
-zs Database Server Option [page 540]

1.4.1.121 -zs Database Server Option

Limits the size of the request log.

> **Syntax**
>
> ```
> dbsrv17 -zs size[ k | m | g ] ...
> ```

Applies to

All operating systems.

Remarks

Request logging is turned on using the -zr option, and redirected to a separate file using the -zo option. You can limit the size of the file using the -zs option.

The `size` is the maximum file size for the request log, in bytes. Use k, m, or g to specify units of kilobytes, megabytes, or gigabytes respectively.

If you specify -zs 0, then there is no maximum size for the request logging file, and the file is never renamed. This is the default value.

When the request log file reaches the size specified by either the -zs option or the sa_server_option system procedure, the file is renamed with the extension `.old` appended (replacing an existing file with the same name if one exists). The request log file is then restarted.

Example

The following example shows how the -zs option is used to control log file size. Suppose you start a database server with the following command line:

```
dbsrv17 -zr all -zs 10k -zo mydatabase.log
```

A new log file `mydatabase.log` is created. When this file reaches 10 KB in size, any existing `mydatabase.old` files are deleted, `mydatabase.log` is renamed to `mydatabase.old`, and a new `mydatabase.log` file is started. This process is repeated each time the `mydatabase.log` file reaches the specified size (in this case 10 KB).

Related Information

Request Logging [page 1502]
-zn Database Server Option [page 535]
-zo Database Server Option [page 536]
-zr Database Server Option [page 538]
sa_server_option System Procedure

1.4.1.122 -zt Database Server Option

Turns on logging of request timing information.

Syntax

```
dbsrv17 -zt ...
```

Applies to

All operating systems.

Remarks

Specifying the -zt option sets the RequestTiming property to On. Once request timing is turned on, you can use the sa_performance_diagnostics system procedure to retrieve all the values of the properties that store request timing information.

The following properties return request timing information only when RequestTiming is turned on. Otherwise the properties return NULL. All of the properties except for ReqStatus are connection, database, and server properties. The database and server properties are an aggregation of the connection values since the database or server started, and these database and server properties only return information if RequestTiming is enabled.

- ReqCountActive connection property
- ReqCountBlockContention connection property
- ReqCountBlockLock connection property
- ReqCountUnscheduled connection property
- ReqStatus connection property
- ReqTimeActive connection property
- ReqTimeBlockContention connection property
- ReqTimeBlockLock connection property
- ReqTimeUnscheduled connection property

Example

You can find the current value of the RequestTiming setting using the following query:

```
SELECT PROPERTY( 'RequestTiming' );
```

Related Information

Request Logging [page 1502]
sa_performance_diagnostics System Procedure
sa_performance_statistics System Procedure

1.4.2 Database Startup Options

The database options are specified on the command line after the database file, and apply only to that database.

In this section:

-ruo Database Option [page 558]
Specifies an offset in the transaction log to restore the database to when performing point-in-time recovery.

-sbx Database Option [page 559]
Controls disk sandboxing for the database.

-sn Database Option [page 560]
Provides an alternate server name for a single database running on a database server.

-wc Database Option [page 561]
Controls whether checksums are enabled on write operations for the database, if it does not have checksums enabled by default.

-xp Database Option [page 563]
Enable database mirroring for a high availability partner or read-only scale-out root or copy node.

1.4.2.1 -a Database Option

Applies the named transaction log.

Syntax

```
dbsrv17 [ server-options ] database-file -a log-filename ...
```

Applies to

All operating systems.

Remarks

This option is used to recover from media failure on the database file. When this option is specified, the database server applies the log and then shuts down. It doesn't continue to run. If you need to apply multiple transaction logs, you must know the correct order in which to apply them when using -a. The database server automatically applies multiple transaction logs in the correct order if you use the -ad or -ar option.

If the specified log file name uses a relative path, then this directory is read relative to the database file directory.

The -a database option must be specified after the `database-file`, and applies only to that database.

Specifying a cache size when starting the server can reduce recovery time.

Example

The following example applies the transaction log file `mydemo.log` to a backup copy of the mydemo database.

```
dbsrv17 "c:\backup\mydemo.db" -a "c:\backup\mydemo.log"
```

Related Information

Database Backup and Recovery [page 939]
Database Recovery with Multiple Transaction Logs [page 979]
Recovering from Media Failure on the Database File [page 984]
-ad Database Option [page 545]
-ar Database Option [page 547]
-as Database Option [page 548]

1.4.2.2 -ad Database Option

Specifies the directories containing transaction log files to be applied to the database.

Syntax

```
dbsrv17 [ server-options ] database-file -ad log-directory-specification ...
```

Applies to

All operating systems.

Remarks

`log-directory-specification` can be one or more directory locations. When specifying more than one directory, use a semicolon as the delimiter.

When you include the -ad option, the specified directories are scanned for transaction log files associated with the database. Appropriate transaction log files with starting log offsets greater than or equal to the start log offset stored in the database file are applied, in log offset order. For point-in-time recovery, changes are applied starting from the time stamp or offset specified.

Once all the transaction log files have been applied, the database is stopped. Specify the -as database server option if you want the database to continue running once the transaction log files have been applied.

If the any of the log directories are specified using a relative path, then this path is interpreted as relative to the `database-file` directory.

The -ad database option must be specified after the `database-file`, and applies only to that database.

Example

The database server applies the transaction log files in the `c:\backup` directory to the `mysample.db` database and then stops the database once the transaction log files have been applied.

```
dbsrv17 "c:\mysample.db" -ad "c:\backup"
```

The database server applies the transaction log files in the `c:\backup` and `c:\backup2` directories to the `mysample.db` database and the database continues running once the transaction log files have been applied.

```
dbsrv17 "c:\mysample.db" -ad "c:\backup;c:\backup2" -as
```

Related Information

Database Recovery with Multiple Transaction Logs [page 979]
Recovering from Media Failure on the Database File [page 984]
-a Database Option [page 544]
-ar Database Option [page 547]
-as Database Option [page 548]
-ru Database Option [page 557]
-ruo Database Option [page 558]

1.4.2.3 -al Database Option

Allows standard user authentication for specified users of the specified database.

Syntax

```
dbsrv17 [ server-options ] database-file -al userid[ ;userid ... ] ...
```

On UNIX and Linux, quotation marks are required for the -al value when semicolons are used.

Applies to

All operating systems.

Remarks

Allow the users specified in the semicolon separated list of user IDs to use standard authentication for the database that the option follows. At most 5 user IDs can be specified.

This option is useful if the login_mode option does not include standard authentication and the user cannot authenticate using alternate means. Users with full DBA privileges can always authenticate using standard authentication.

Related Information

login_mode Option [page 752]

1.4.2.4 -ar Database Option

Specifies that any transaction log files located in the same directory as the current transaction log should be applied to the database.

Syntax

```
dbsrv17 [ server-options ] database-file -ar  ...
```

Applies to

All operating systems.

Remarks

When you include the -ar option, the database server looks for transaction log files associated with the database that are located in the same directory as the current transaction log. The transaction log location is obtained from the database. Transaction log files with starting log offsets greater than or equal to the start log offset stored in the database are applied, in log offset order. Once all the transaction log files have been applied, the database is stopped. You must also specify the -as option if you want the database to continue running once the transaction log files have been applied.

The -ar database option must be specified after the `database-file`, and applies only to that database.

Example

The database server applies the transaction log files (whose location is obtained from the database) to the `mysample.db` database. The database continues running after the transaction log files have been applied.

```
dbsrv17 "c:\mysample.db" -ar -as
```

Related Information

Database Recovery with Multiple Transaction Logs [page 979]
Recovering from Media Failure on the Database File [page 984]
-a Database Option [page 544]
-ad Database Option [page 545]
-as Database Option [page 548]

1.4.2.5 -as Database Option

Specifies that the database should continue to run after transaction logs have been applied (used in conjunction with -ad or -ar).

Syntax

```
dbsrv17 [ server-options ] database-file { -ad log-dir | -ar } -as  ...
```

Applies to

All operating systems.

Remarks

The -as option must be specified in conjunction with either the -ad or -ar option. When you include -as, the database continues running after the transaction logs are applied to it.

The -as database option must be specified after the `database-file`, and applies only to that database.

Example

The database server applies the transaction log files to the `mysample.db` database. In this case, because -ar is specified, the database server obtains the location of the transaction logs from the database. The database continues running after the transaction log files have been applied.

```
dbsrv17 "c:\mysample.db" -ar -as
```

The database server applies the transaction log files in the `backup` directory to the `mysample.db` database. The database continues running after the transaction log files have been applied.

```
dbsrv17 "c:\mysample.db" -ad "c:\backup" -as
```

Related Information

Database Recovery with Multiple Transaction Logs [page 979]
Recovering from Media Failure on the Database File [page 984]
-a Database Option [page 544]
-ad Database Option [page 545]
-ar Database Option [page 547]

1.4.2.6 -ds Database Option

Specifies the directory where the dbspaces for the database and the transaction log are located.

Syntax

```
dbsrv17 database-file -ds dbspace-directory ...
```

Applies to

All operating systems.

Remarks

When a dbspace directory is specified, the database server only searches this directory for dbspaces. The location of the dbspace appears in the database server messages window.

If your backup includes dbspaces with full path names, you can use this option to start the backed up copy of the database on the same computer as the original database while the original database is still running.

The -ds database option must be specified after the `database-file`, and applies only to that database.

If a transaction log file is not found in the directory specified by this option, then one is created in this location.

> ⚠ Caution
>
> The -ds option should only be used for recovery. If you specify this option and the database has a current, live transaction log that is not located in the directory specified by the -ds option, then a new transaction log is created in the specified location.

Example

The following example starts a database server that looks for dbspaces in the directory `c:\backup\Nov15`:

```
dbsrv17 c:\backup\Nov15\my.db -ds c:\backup\Nov15\
```

The following example starts a database server that looks for dbspaces in the current directory:

```
dbsrv17 my.db -ds .
```

Related Information

Additional Dbspaces Considerations [page 283]
START DATABASE Statement
STOP DATABASE Statement
default_dbspace Option [page 723]

1.4.2.7 -dh Database Option

Prevents this database from appearing when the Server Enumeration utility (dblocate) is used against this server.

> Syntax
>
> ```
> dbsrv17 [ server-options ] database-file -dh ...
> ```

Applies to

All operating systems.

Remarks

The -dh option makes a database undetectable when the Server Enumeration utility (dblocate) is run against the server. Therefore, when dblocate is used with the -d option, the -dn option, or the -dv option, the database isn't listed.

The -dh database option must be specified after the `database-file`, and applies only to that database.

Related Information

Server Enumeration Utility (dblocate) [page 1200]

1.4.2.8 -ek Database Option

Specifies the key for a strongly encrypted database.

Syntax

```
dbsrv17 [ server-options ] database-file -ek key ...
```

Applies to

All operating systems.

Remarks

You must provide the key value with the -ek option to start an encrypted database. The key is a string, including mixed cases, numbers, letters, and special characters.

To enter the encryption key in a window so it cannot be seen in clear text, use the -ep server option.

To secure communication packets between client applications and the database server, use the -ec server option and transport layer security.

The -ek database option must be specified after the `database-file`, and applies only to that database.

Example

The following example starts a database and specifies the encryption key on the command line.

```
dbsrv17 -x tcpip mydata.db -ek "Akmm9u70y"
```

Related Information

Database Encryption and Decryption [page 1697]
Transport Layer Security [page 1727]
-ec Database Server Option [page 418]
-ep Database Server Option [page 423]
DatabaseKey (DBKEY) Connection Parameter [page 70]

1.4.2.9 -f Database Option

Forces the database server to start after the transaction log has been lost.

Syntax

```
dbsrv17 -f ...
```

Applies to

All operating systems.

Remarks

⚠ Caution

This option is for use in recovery situations only.

If there is no transaction log, the database server performs a checkpoint recovery of the database and then shuts down. It doesn't continue to run. You can then restart the database server without the -f option for normal operation.

If there is a transaction log in the same directory as the database, the database server performs a checkpoint recovery, and a recovery using the transaction log, and then shuts down. It doesn't continue to run. You can then restart the database server without the -f option for normal operation.

Specifying a cache size when starting the server can reduce recovery time.

Example

The following command forces the database server to start and perform a recovery of the database `mydatabase.db`:

```
dbsrv17 mydatabase.db -f
```

Related Information

Special Modes [page 341]
Database Backup and Recovery [page 939]

1.4.2.10 -m Database Option

Truncates the transaction log when a checkpoint is performed.

Syntax

```
dbsrv17 [ server-options ] database-file -m ...
```

Applies to

All operating systems.

Remarks

Truncates the transaction log when a checkpoint is done, either at shutdown or as a result of a checkpoint scheduled by the server. This option provides a way to limit the growth of the transaction log automatically. Checkpoint frequency is still controlled by the checkpoint_time and recovery_time options (or the -gc and -gr database server options).

The -m option is useful where high volume transactions requiring fast response times are being processed, and the contents of the transaction log aren't being relied upon for recovery or replication. When this option is selected, there is no protection against media failure on the device that contains the database files.

To avoid database file fragmentation, place the transaction log on a separate device or partition from the database itself.

This option is the same as the -m server option, but applies only to the current database or the database identified by the `database-file` variable.

> ⚠ Caution
>
> Do not use the -m option with databases that are being replicated or synchronized.
>
> Replication and synchronization, used by SQL Remote and MobiLink, inherently rely on transaction log information.

The -m database option must be specified after the `database-file`, and applies only to that database.

Example

The following example starts a database server named silver and loads the database `salesdata.db`. When a checkpoint is done, the transaction log contents are deleted.

```
dbsrv17 -n silver "c:\inventory details\salesdata.db" -m
```

Related Information

The Transaction Log [page 292]
-m Database Server Option [page 465]
Transaction Log Utility (dblog) [page 1231]

1.4.2.11 -n Database Option

Sets the name of the database.

> Syntax
>
> ```
> dbsrv17 [ server-options ] database-file -n string ...
> ```

Default

Database receives the name of the database file with the path and extension removed

Applies to

All operating systems.

Remarks

Both database servers and databases can be named. Since a database server can load several databases, the database name is used to distinguish the different databases.

Database names cannot:

- begin with white space, single quotes, or double quotes
- end with white space
- contain semicolons
- be longer than 250 bytes

The -n database option must be specified after the `database-file`, and applies only to that database.

You can only use the database name utility_db to connect to the utility database.

> **i** Note
>
> There are two -n options. The -n option is position-dependent. If it appears before a database file name, it is a server option and names the server. If it appears after a database file name, it is a database option and names the database.
>
> For example, the following command names the server SERV and the database DATA:
>
> ```
> dbsrv17 -n SERV c:\mydata.db -n DATA
> ```

Example

If the database that is started is *%SQLANYSAMP17%\demo.db* and no -n option is specified, the name of the database is demo.

The following example starts the database server with a cache size of 3 MB, loads the database, and names the database test. Since no database server name has been specified, the server takes its name from the first database, so the server's name is also test.

```
dbsrv17 -c 3MB "c:\mydata.db" -n "test"
```

Related Information

Database Server Names and Database Names [page 334]
The Utility Database (utility_db) [page 309]
-n Database Server Option [page 467]

1.4.2.12 -r Database Option

Starts the named database as read-only.

Syntax

```
dbsrv17 [ server-options ] database-file -r ...
```

Applies to

All operating systems.

Remarks

Opens all database files (the main database file, dbspaces, transaction log, and transaction log mirrors) as read-only, with the exception of the temporary file, when the option is specified before any database names on the command line. No changes to the database(s) are allowed: the database server does not modify the database file(s) and transaction log files.

If the -r option is specified after a database file, only that specific database is read-only. You can make changes on temporary tables, but ROLLBACK has no effect, since the transaction and rollback logs are disabled.

A database distributed on a CD-ROM device is an example of a database file that cannot be modified. You can use read-only mode to access this sort of database.

If you attempt to modify the database, for example with an INSERT or DELETE statement, a SQLSTATE_READ_ONLY_DATABASE error is returned.

Databases that require recovery cannot be started in read-only mode. For example, database files created using an online backup cannot be started in read-only mode if there were any open transactions when the backup was started, since these transactions would require recovery when the backup copy is started.

You cannot start a database in read-only mode if auditing is turned on.

Example

To open two databases in read-only mode, run the following:

```
dbsrv17 -r database1.db database2.db
```

To open only the first of two databases in read-only mode, run the following:

```
dbsrv17 database1.db -r database2.db
```

Related Information

-r Database Server Option [page 488]
auditing Option [page 688]

1.4.2.13 -ru Database Option

Specifies a timestamp to restore the database to when performing point-in-time recovery.

Specifying a time zone in the timestamp specification is optional, but essential when the database server and the transaction log are in different time zones. Set the time zone specification to match the time zone of the location where the transaction log is located.

Syntax

```
dbsrv17 [ server-options ] database-file -ru "time-stamp [ time-zone ]" ...
```

Applies to

All operating systems.

Remarks

Including the time zone (for example, -05:00) in the timestamp specification is a best practice and recommended when doing point-in-time recovery to a timestamp. When you omit the time zone in the timestamp specification, the local time of the database server doing the restore is used. If the timezone of the database server performing the restore matches the timezone used when the transaction logs were generated, then there is no problem.. However, if the timezones do not match, then the database is restored to the specified time in the timezone of the database server, which is not correct.

.

Example

The following command restores the myDB.db database to the timestamp 2014-12-15 15:23:15 in the time zone UTC minus 5 hours (-05:00) in the transaction log:

```
dbsrv17 myDB.db -as -ar -ru "2014-12-15 15:23:15 -05:00"
```

Related Information

Database Recovery [page 960]
Point-in-time Recovery [page 962]
Tutorial: Restoring the Database to a Timestamp [page 965]
-ad Database Option [page 545]
-ruo Database Option [page 558]

1.4.2.14 -ruo Database Option

Specifies an offset in the transaction log to restore the database to when performing point-in-time recovery.

The transaction log sequentially numbers the operations and events it records; these numbers are called **offsets**.

Syntax

```
dbsrv17 [ server-options ] database-file -ruo offset-number ...
```

Applies to

All operating systems.

Remarks

Operations are recorded sequentially in the transaction log, thus every operation can be uniquely identified by a numeric offset within the file. If you can identify an operation or event in the log up to which, and including which, the database was fine, then you can restore the database up to and including the offset of this operation.

Example

The following command restores the myDB.db database to the offset 1245672 in the transaction log:

```
dbsrv17 myDB.db -as -ar -ruo 1245672
```

Related Information

Database Recovery [page 960]
Point-in-time Recovery [page 962]
Tutorial: Restoring the Database to an Offset in the Transaction Log [page 970]
-ad Database Option [page 545]
-ru Database Option [page 557]

1.4.2.15 -sbx Database Option

Controls disk sandboxing for the database.

Syntax

```
dbsrv17 [ server-options ] database-file -sbx{ + | - } ...
```

Applies to

All operating systems.

Allowed Values

-sbx+ Enable disk sandbox for this database.
-sbx Enable disk sandbox for this database.
-sbx- Disable disk sandbox for this database.

Remarks

Enabling this option restricts the read-write file operations on the database to the directory where the main database file is located and any subdirectories of this directory.

If this option is not specified, the default disk sandbox settings specified by the -sbx database server option are used.

You can temporarily change the disk sandbox settings for the database while the database is running by using the DiskSandbox option in the sa_db_option system procedure.

Related Information

disk_sandbox Option [page 730]
-sbx Database Server Option [page 491]
sa_db_option System Procedure

1.4.2.16 -sn Database Option

Provides an alternate server name for a single database running on a database server.

Syntax

```
dbsrv17 [ server-options ] database-file -sn alternate-server-name ...
```

Applies to

All operating systems.

Network database server (dbsrv17) only.

Remarks

The -sn database option must be specified after the `database-file`, and applies only to that database.

The database server can be configured to listen for more than one server name for a particular database server. Server names other than the real server name are called alternate server names, and are specific to a particular database running on the database server. Clients can specify the alternate server name using the ServerName (Server) connection parameter to connect to that database.

Alternate server names must be unique on the network. If you attempt to start a database on the command line and the alternate server name you specify is not unique, the database server fails to start. You can also provide an alternate server name using the START DATABASE statement.

Clients that specify an alternate server name can only connect to the database that specified the alternate server name. They cannot connect to, create, stop, and drop other databases on the same database server. If the DBN or DBF connection parameter is specified, it must match the database name or database file,

respectively. If the DBN or DBF connection parameter is not specified, then the database acts as the default database for that server.

The Server Enumeration utility (dblocate) detects alternate server names.

Example

The following command starts the databases `satest.db` and `sample.db` on a database server named myserver. The -sn option instructs the database server to use mysample as an alternate server name when connecting to `sample.db`.

```
dbsrv17 -n myserver satest.db sample.db -sn mysample
```

You can connect to `sample.db` using any of the following connection parameters:

- Server=myserver;DBN=sample
- Server=mysample
- Server=mysample;DBN=sample

You cannot connect to `satest.db` using Server=mysample.

Related Information

Separately Licensed Components
Database Mirroring [page 1757]
CREATE MIRROR SERVER Statement
START DATABASE Statement
Server Enumeration Utility (dblocate) [page 1200]

1.4.2.17 -wc Database Option

Controls whether checksums are enabled on write operations for the database, if it does not have checksums enabled by default.

Syntax

```
dbsrv17 [ server-options ] database-file -wc[ + | - ] ...
```

Applies to

All operating systems.

Allowed Values

-wc+ Enable checksums for this database.
-wc Disable checksums for this database if it does not have checksums automatically enabled.
-wc- Disable checksums for this database if it does not have checksums automatically enabled.

Remarks

The difference between the write checksums (enabled with the -wc option) and global checksums (creating a database with checksums enabled) is that with -wc, database pages are checksummed only when they are written out to disk. Pages that are read from disk are only verified if a checksum value was calculated before the pages were written. If a database has checksums enabled, checksums are calculated for all pages when they are written and checksums are verified for all pages when they are read.

If the database server detects that the database is running on a removable storage device, such as a network share or USB device, then the database server automatically enables global checksums for all database pages.

By default, databases created with version 10 and 11 of SQL Anywhere do not have global checksums enabled. If you start a database created with SQL Anywhere 11 on a version 12 or later database server, then by default the database server creates write checksums for pages when they are written to disk (-wc+). Version 12 and later databases have global checksums enabled by default, so the database server defaults to -wc- for these databases because by default all database pages have checksums. You can use either the -wc option or the START DATABASE statement to change the database server's checksum behavior if you do not want to use the default checksum settings.

You can check whether a database was created with global checksums enabled by executing the following statement:

```
SELECT DB_PROPERTY ( 'Checksum' );
```

You can check whether write checksums are enabled for write I/O operations only by executing the following statement:

```
SELECT DB_PROPERTY ( 'WriteChecksum' );
```

Related Information

Corruption Detection Using Checksums [page 993]
-wc Database Server Option [page 522]
START DATABASE Statement

1.4.2.18 -xp Database Option

Enable database mirroring for a high availability partner or read-only scale-out root or copy node.

Syntax

```
dbsrv17 [ server-options ] database-file -xp on ...
```

Applies to

All operating systems.

Network database server (dbsrv17) only.

Remarks

The -xp database option must be specified after the `database-file`, and applies only to that database. You cannot start a database that doesn't have a transaction log in mirroring mode.

on

You can only use database mirroring and/or scale-out if you specify the -xp option when the database server is started, even if mirroring or scale-out information is stored in the database. The value off is not supported. Database mirroring and scale-out settings are defined in the database by using the following statements:

- CREATE MIRROR SERVER
- SET MIRROR OPTION

When you specify **`-xp on`**, also specify the name of the database server in the mirroring system by using the -n database option. Include the *-su* option to specify the password for the utility database. Then, you can use the utility database to shut down the database server or force the mirror server to become the primary server, if necessary.

Example

The following command starts three database files on a database server that is available to participate in a mirroring system:

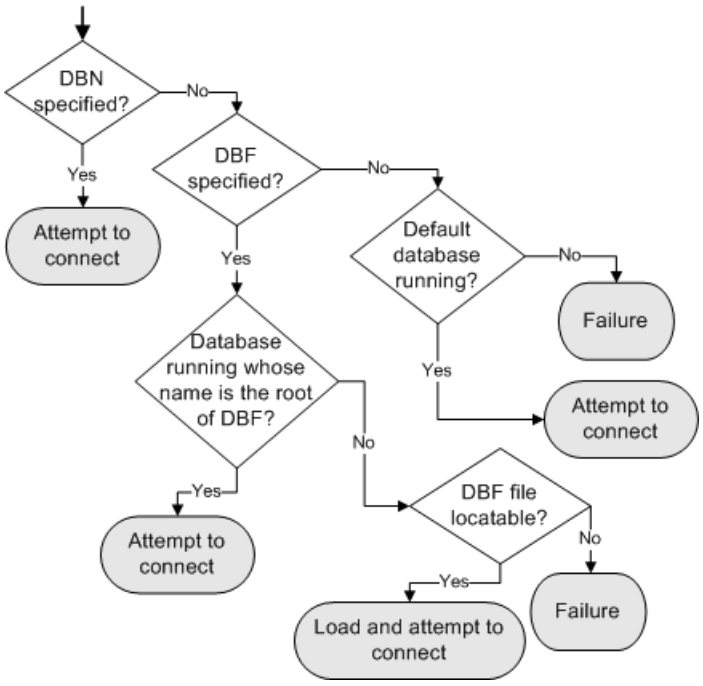
```
dbsrv17 -n server1 -x tcpip(PORT=6871) -su passwd
    c:\server1\one.db -xp on c:\server1\two.db -xp on c:\server1\three.db -xp on
```

Related Information

Read-only Scale-out [page 1830]
Separately Licensed Components
Database Mirroring Modes [page 1768]
Preferred Database Server in a Database Mirroring System [page 1774]
Database Mirroring Modes [page 1768]
Setting up a Database Mirroring System [page 1771]
Tutorial: Creating a Read-only Scale-out System [page 1853]
Tutorial: Creating a Database Mirroring System [page 1800]
Tutorial: Creating a Database Mirroring System with Multiple Databases That Share an Arbiter Server [page 1808]
CREATE MIRROR SERVER Statement
SET MIRROR OPTION Statement
-su Database Server Option [page 502]
-xa Database Server Option [page 525]
-xf Database Server Option [page 527]
-n Database Server Option [page 467]

1.4.3 Troubleshooting: How Database Servers Are Located

When the client successfully locates a server, it then tries to locate the database.



1.5 Database Configuration

Several tools and methods are provided for configuring a database.

In this section:

1.5.1 Environment Variables

Environment variables are used to store various types of information.

Not all environment variables must be set in all circumstances.

Setting Environment Variables on Windows

The SQL Anywhere installer creates or modifies the following environment variables in your computer's properties: PATH and SQLANY17.

You can view the environment variables set for a database server by starting the database server with the -ze option. After the database server is started, you can view environment variables using the xp_getenv system procedure.

Other environment variables can be set by modifying the properties for your computer, or within command prompts or batch files by using the SET command.

In this section:

Specifies the location of the directory containing SQL Anywhere 17.

SQLANYSAMP17 Environment Variable [page 581]
Specifies the location of the SQL Anywhere samples directory.

SQLCONNECT Environment Variable [page 582]
Specifies additional connection parameters used when connecting to the database server.

SQLPATH Environment Variable [page 583]
Specifies the location of SQL script and Help files.

SQLREMOTE Environment Variable [page 583]
Specifies the directory that contains addresses for the SQL Remote FILE message link.

SYBASE Environment Variable [page 584]
Specifies the home directory for the installation of some SAP applications, including SAP Adaptive Server Enterprise, SAP Open Client, SAP Open Server, and utilities such as DSEdit.

TMP, TMPDIR, and TEMP Environment Variables [page 585]
Specify the location of SQL Anywhere temporary files.

1.5.1.1 UNIX and Linux Environment Variables

Once the software is installed, each user must set some environment variables for the system to locate and run applications.

The installer creates special files to help set some environment variables for the system to locate and run applications: `sa_config` and `sample_config.sh` These files are installed in *$SQLANY17*`/bin32` and *$SQLANY17*`/bin64`. Each file sets needed user environment variables.

There are `.sh` and `.csh` variants of these files. As these file extensions imply, one type is designed to work under Bourne shell (`sh`) and its derivatives (such as `ksh` or `bash`). The other type is designed to work under C-shell (`csh`) and its derivatives (such as `tcsh`).

Some statements are commented out in each of these files. The system administrator may want to edit these files and remove comments, depending on the configuration of their system.

To run a SQL Anywhere application, you have several choices:

1. If you add the environment variables from these files to your system environment, you can run applications by launching them from a GUI, such as X window server, or by typing the application name in a terminal window.
2. In a terminal window, if you source one of the files, you can run an application by typing its name.
3. *$SQLANY17*`/bin32s` and *$SQLANY17*`/bin64s` contain scripts with the same names as SQL Anywhere applications. These scripts set the appropriate environment variables before launching the application. You can run the application by running the corresponding script. You do not have to source these files before you run these scripts.

Setting Environment Variables for the Finder on macOS

The `sa_config` file sets the following environment variables: DYLD_LIBRARY_PATH, PATH, and SQLANY17. The `sample_config` file sets the ODBCINI environment variable. Rebooting is not required.

Terminal sessions do not inherit environment variables from the Finder.

Related Information

Sourcing sa_config.sh, sample_config32.sh, and sample_config64.sh [UNIX/Linux] [page 568]

1.5.1.2 Sourcing sa_config.sh, sample_config32.sh, and sample_config64.sh [UNIX/Linux]

Source the `sa_config.sh/sa_config.csh` and `sample_config32.sh/sample_config64.sh` files.

Context

Before you can use the samples, you must run the configuration `sample_config32.sh` or `sample_config64.sh`.

Procedure

1. Source the SQL Anywhere environment variables using one of the supplied scripts.

   Bourne shell and its derivatives

   Under Bourne shell and its derivatives, the name of this command is `.` (a single period). For example, if SQL Anywhere is installed in */opt/sqlanywhere17*, the following statement sources `sa_config.sh` (use `bin32` for 32-bit environments):

   ```
   .  /opt/sqlanywhere17/bin64/sa_config.sh
   ```

   For example, on a Mac, run the following command to source `sa_config.sh`:

   ```
   . /Applications/SQLAnywhere17/System/bin64/sa_config.sh
   ```

   C-shell and its derivatives

Under C-shell and its derivatives, the command is `source`. For example, if SQL Anywhere is installed in */opt/sqlanywhere17*, the following command sources `sa_config.csh` (use `bin32` for 32-bit environments):

```
source  /opt/sqlanywhere17/bin64/sa_config.csh
```

2. Configure the samples. Before using the samples for SQL Anywhere 17, run the configuration script corresponding to the bitness of SQL Anywhere you want to use with them.

   For example, to configure the samples for use with 32-bit software, run:

   ```
   . /opt/sqlanywhere17/samples/sample_config32.sh
   ```

   For example, on a Mac, to configure the samples for use with a 32-bit software, run:

   ```
   . /Applications/SQLAnywhere17/samples/sample_config32.sh
   ```

Results

The files are sourced.

1.5.1.3 DYLD_LIBRARY_PATH Environment Variable [macOS]

Specifies the directories that are searched at run time for libraries required by SQL Anywhere applications on macOS.

Syntax

```
DYLD_LIBRARY_PATH=path-list
```

Default

/Applications/SQLAnywhere17/System`/lib64`*:/Applications/SQLAnywhere17/System*`/lib32`

Remarks

The `sa_config.sh` and `sa_config.csh` files, created by the installer, are scripts that create or modify this and other environment variables.

Related Information

UNIX and Linux Environment Variables [page 567]
LD_LIBRARY_PATH Environment Variable [Linux and Solaris] [page 570]
LIBPATH Environment Variable [IBM AIX] [page 571]
SHLIB_PATH Environment Variable [HP-UX] [page 579]

1.5.1.4 LD_LIBRARY_PATH Environment Variable [Linux and Solaris]

Specifies the directories that are searched at run time for libraries required by SQL Anywhere applications on Linux and Solaris.

Syntax

```
LD_LIBRARY_PATH=path-list
```

Default

32-bit Linux

```
$SQLANY17/lib32
$SQLANY17/lib64
$SQLANY17/bin32/jre180/lib/i386/server
```

64-bit Linux and Solaris

```
$SQLANY17/lib64
$SQLANY17/lib32
$SQLANY17/bin64/jre180/lib/amd64/server
$SQLANY17/bin64/jre180/lib/amd64
```

Remarks

The `sa_config.sh` and `sa_config.csh` files, created by the installer, are scripts that create or modify this and other environment variables.

Related Information

UNIX and Linux Environment Variables [page 567]

DYLD_LIBRARY_PATH Environment Variable [macOS] [page 569]
LIBPATH Environment Variable [IBM AIX] [page 571]
SHLIB_PATH Environment Variable [HP-UX] [page 579]

1.5.1.5 LIBPATH Environment Variable [IBM AIX]

Specifies the directories that are searched at run time for libraries required by SQL Anywhere applications on IBM AIX.

Syntax

```
LIBPATH=path-list
```

Default

- `/usr/lpp/sqlanywhere17/lib32` (32-bit platforms)
- `/usr/lpp/sqlanywhere17/lib64` (64-bit platforms)

Remarks

The `sa_config.sh` and `sa_config.csh` files, created by the installer, are scripts that create or modify this and other environment variables.

Related Information

UNIX and Linux Environment Variables [page 567]
DYLD_LIBRARY_PATH Environment Variable [macOS] [page 569]
LD_LIBRARY_PATH Environment Variable [Linux and Solaris] [page 570]
SHLIB_PATH Environment Variable [HP-UX] [page 579]

1.5.1.6 MLTEMPLATE_JARS Environment Variable

Specifies the location of a JDBC driver. The environment variable is used by MobiLink Template utility (mltemplate).

Syntax

```
MLTEMPLATE_JARS=path-to-JDBC-driver
```

Remarks

This environment variable is used by the mltemplate utility to import the database schema of a non-SQL Anywhereconsolidated database.

Specify the path to the JAR file that contains the JDBC driver of the non-SQL Anywhere database.

You do not need to provide the path for the SQL Anywhere JDBC driver.

Related Information

MobiLink Template Utility (mltemplate)

1.5.1.7 ODBCHOME Environment Variable [UNIX/Linux]

Specifies the location of the `.odbc.ini` file.

Syntax

```
ODBCHOME=odbc-ini-directory
```

Remarks

The `.odbc.ini` file is the system information file that contains ODBC data sources. If the file is named anything other than `.odbc.ini`, you must use the ODBCINI or ODBC_INI environment variable to specify its location.

Related Information

UNIX and Linux Environment Variables [page 567]
ODBC Data Sources on UNIX/Linux [page 126]
ODBCINI and ODBC_INI Environment Variables [UNIX/Linux] [page 573]

1.5.1.8 ODBCINI and ODBC_INI Environment Variables [UNIX/Linux]

Specifies the path and name of the system information file containing ODBC data sources.

Syntax

```
ODBCINI=odbc-ini-file
```

```
ODBC_INI=odbc-ini-file
```

Remarks

The file name does not need to be `.odbc.ini` if it is specified using one of these environment variables. Both environment variables are provided for compatibility with other products.

Related Information

UNIX and Linux Environment Variables [page 567]
ODBC Data Sources on UNIX/Linux [page 126]
ODBCHOME Environment Variable [UNIX/Linux] [page 572]

1.5.1.9 PATH Environment Variable

Specifies the locations of directories containing SQL Anywhere executables.

Syntax

```
PATH=path-list
```

Default

The following paths are only added if the corresponding component is installed.

Operating system	Default location
Windows (32-bit)	*C:\Program Files\SQL Anywhere 17*\bin32
Windows (64-bit)	*C:\Program Files\SQL Anywhere 17*\bin64;*C:\Program Files\SQL Anywhere 17*\bin32
macOS (64-bit)	*/Applications/SQLAnywhere17/System*/bin64
IBM AIX (64-bit)	/usr/lpp/sqlanywhere*17*/bin64
Solaris Sparc (64-bit)	*/opt/sqlanywhere17*/bin64/jre180/bin/sparcv9:*/opt/sqlanywhere17*/bin64:*/opt/sqlanywhere17*/bin32
Linux (32-bit)	*/opt/sqlanywhere17*/bin32/jre180/bin:*/opt/sqlanywhere17*/bin32
Linux (64-bit)	*/opt/sqlanywhere17*/bin64/jre180/bin:*/opt/sqlanywhere17*/bin64:*/opt/sqlanywhere17*/bin32
Other Unix operating systems (64-bit)	*/opt/sqlanywhere17*/bin64

Remarks

On Windows, the PATH environment variable is modified by the installer to include the directories where SQL Anywhere executables are located.

On Unix, the `sa_config.sh` and `sa_config.csh` files, created by the installer, are scripts that create or alter this and other environment variables.

Related Information

UNIX and Linux Environment Variables [page 567]
Sourcing sa_config.sh, sample_config32.sh, and sample_config64.sh [UNIX/Linux] [page 568]

1.5.1.10 SACHARSET Environment Variable

Specifies the default character set encoding used by SQL Anywhere.

Syntax

```
SACHARSET=charset-label
```

Remarks

The `charset-label` is a character set encoding.

You can use the dbinit -le option to obtain a list of all of the available character set encodings for a database.

If SACHARSET is not specified, the operating system character set encoding is used as the default.

Not all components use the SACHARSET environment variable. It is used by Embedded SQL and ODBC connections. Database tools like the Initialization utility (dbinit) use it. It is not used by JDBC connections, including Interactive SQL (dbisql). It is not used by .NET connections.

> **i** Note
>
> Setting SACHARSET is not equivalent to specifying the CharSet (CS) connection parameter for client/database connections.

Related Information

UNIX and Linux Environment Variables [page 567]
Recommended Character Sets and Collations [page 634]
Initialization Utility (dbinit) [page 1166]

1.5.1.11 SADIAGDIR Environment Variable

Specifies the location of the SQL Anywhere diagnostic directory.

> Syntax
>
> ```
> SADIAGDIR=diagnostic-information-directory
> ```

Default

This environment variable is not set by default.

Remarks

This environment variable overrides the default location for the SQL Anywhere diagnostic directory. SQL Anywhere programs use this directory to store reports if the software has a fatal error.

SQL Anywhere stores crash reports and feature statistics information in the diagnostic directory. The SADIAGDIR environment variable is used to determine the location of the diagnostic directory where SQL Anywhere writes crash reports.

If the directory specified by this environment variable does not exist, then the database server operates as though the environment variable is not set.

For additional information on defaults, see "Support Utility" in related information.

Related Information

UNIX and Linux Environment Variables [page 567]
Troubleshooting: Reporting an Error [page 1021]
Support Utility (dbsupport) [page 1221]

1.5.1.12 SALANG Environment Variable

Specifies the language code for the SQL Anywhere database server and tools.

Syntax

```
SALANG=language-code
```

Remarks

The `language-code` is a two-letter combination representing a language. For example, setting *SALANG*=DE sets the default language to German.

The first of the following methods that returns a value determines the default language for message text:

1. Check the SALANG environment variable.
2. (Windows) Check the registry as set during installation or by `dblang.exe`.
3. Query the operating system for language information.
4. If no language information is set, English is the default.

Related Information

Locale Language [page 614]
UNIX and Linux Environment Variables [page 567]
Language Selection Utility (dblang) [page 1186]
Registry Settings on Installation [page 592]

1.5.1.13 SALOGDIR Environment Variable

Specifies the location of the `backup.syb` file.

Syntax

```
SALOGDIR=directory-name
```

Remarks

If the SALOGDIR environment variable is set, it is assumed to contain the path for a directory where the backup history file, `backup.syb` can be written. This file is updated each time you execute a BACKUP or RESTORE statement.

On Windows, the `backup.syb` file is written in the first writable location in the following list:

1. The SALOGDIR environment variable.
2. `%ALLUSERSPROFILE%\SQL Anywhere 17`.
3. The root directory of the current drive.
4. The current directory.

On Windows CE, the `backup.syb` file is written in the first writable location in the following list:

1. The SALOGDIR environment variable.
2. The directory where the database server was started.
3. The root directory.
4. The current directory.

On Unix, the `backup.syb` file is written in the first writable location in the following list:

1. The SALOGDIR environment variable.
2. `$HOME/.sqlanywhere17`.
3. The current directory.

Related Information

UNIX and Linux Environment Variables [page 567]
BACKUP DATABASE Statement

1.5.1.14 SATMP Environment Variable

Specifies the location of temporary files used by the database server and the SQL Anywhere command line utilities that require a temporary directory.

Syntax

```
SATMP=directory-name
```

Remarks

SQL Anywhere creates two types of temporary files: database server-related temporary files (created on all platforms), and communications-related temporary files (created only on UNIX and Linux for both the client and the server).

The SATMP environment variable specifies the location of temporary files used by the database server and the SQL Anywhere command line utilities that require a temporary directory. It is useful when running the database server as a service because it enables you to hold the temporary file in a directory that cannot be accessed by other programs.

If the location of the temporary file is not specified with the -dt option when the database server is started, then the database server checks the value of the SATMP environment variable to determine where to place the temporary file. If the SATMP environment variable does not exist, then the first of the TMP, TMPDIR, or TEMP environment variables to exist is used. On Microsoft Windows, if none of the above environment variables exist, the current directory is used. On UNIX and Linux, if none of the above environment variables exist, `/tmp` is used.

On UNIX and Linux, both the client and the database server must set SATMP to the same value when connecting via shared memory.

Note

In SQL Anywhere version 9 and earlier, the environment variable ASTMP is equivalent to SATMP. If you are using shared memory to connect version 9 and version 10 software, you must set the SATMP and ASTMP environment variables to specify the (same) location of the temporary directory.

To restrict the permissions of the temporary files created by the database server or client on UNIX and Linux, you must set this environment variable to a directory that is not in the following list:

- `/tmp`
- `/tmp/.SQLAnywhere`
- the value of the TMP environment variable, if set
- the value of the TMPDIR environment variable, if set
- the value of the TEMP environment variable, if set
- a symbolic link pointing to any of the above directories

When SATMP is set to a directory that is not listed above, the database server searches up the given directory path looking for directories owned by the current user with permissions set to 770, 707, or 700. If the

permissions are not set to one of these values, files are created with permissions set to 777. For each directory that is found, the database server removes the appropriate permissions (other, group, and other+group, respectively) from the permission mask used to create temporary files.

> ⚠ Caution
>
> Setting SATMP to a directory that is not included in the list above may affect the ability of users using different UNIX or Linux accounts to connect to the database server over shared memory.

On UNIX, you can set the SATMP environment variable to a unique directory when securing shared memory connections.

Related Information

UNIX and Linux Environment Variables [page 567]
-dt Database Server Option [page 416]
TMP, TMPDIR, and TEMP Environment Variables [page 585]
Troubleshooting: Database Server Startup [page 1023]
sa_disk_free_space System Procedure

1.5.1.15 SHLIB_PATH Environment Variable [HP-UX]

Specifies the directories that are searched at run time for libraries required by SQL Anywhere applications on HP-UX.

> ⁖ Syntax
>
> ```
> SHLIB_PATH=path-list
> ```

Default

- */opt/sqlanywhere17*`/lib32` (32-bit platforms)
- */opt/sqlanywhere17*`/lib64` (64-bit platforms)

Remarks

The `sa_config.sh` and `sa_config.csh` files, created by the installer, are scripts that create or modify this and other environment variables.

Related Information

UNIX and Linux Environment Variables [page 567]
DYLD_LIBRARY_PATH Environment Variable [macOS] [page 569]
LD_LIBRARY_PATH Environment Variable [Linux and Solaris] [page 570]
LIBPATH Environment Variable [IBM AIX] [page 571]

1.5.1.16 SQLANY17 Environment Variable

Specifies the location of the directory containing SQL Anywhere 17.

Syntax

```
SQLANY17=directory-name
```

Default

Operating system	Location
Microsoft Windows	*C:\Program Files\SQL Anywhere 17*
IBM AIX	`/usr/lpp/sqlanywhere`*17*
macOS	*/Applications/SQLAnywhere17/System*
Other UNIX and Linux operating systems	*/opt/sqlanywhere17*

Remarks

This environment variable should be set for several reasons. For example, samples require this environment variable to locate SQL Anywhere applications.

On Microsoft Windows, the installer creates the SQLANY17 environment variable (it is stored in the Microsoft Windows Registry).

On UNIX and Linux, the `sa_config.sh` and `sample_config32.csh`/ files, created by the installer, are scripts that create or modify this and other environment variables.

Related Information

UNIX and Linux Environment Variables [page 567]

Sourcing sa_config.sh, sample_config32.sh, and sample_config64.sh [UNIX/Linux] [page 568]

1.5.1.17 SQLANYSAMP17 Environment Variable

Specifies the location of the SQL Anywhere samples directory.

Syntax

```
SQLANYSAMP17=directory-name
```

Default

Operating system	Default location
Microsoft Windows	*C:\Users\Public\Documents*`\SQL Anywhere 17\Samples`
UNIX and Linux (single-user installations)	*$SQLANY17*`/samples`

Remarks

On Microsoft Windows, the installer creates the %SQLANYSAMP17% environment variable (it is stored in the Microsoft Windows Registry). To open a Microsoft Windows Explorer window in the SQL Anywhere samples directory, click *Start* > *All Programs* > *SQL Anywhere 17* > *Sample Applications And Projects*.

On UNIX and Linux, including macOS, source the `sample_config32.sh` or the `sample_config64.sh` script to create a per-user copy of the SQL Anywhere samples and set the location of the $SQLANYSAMP17 environment variable to the newly created per-user copy.

Related Information

UNIX and Linux Environment Variables [page 567]
SQLANY17 Environment Variable [page 580]

1.5.1.18 SQLCONNECT Environment Variable

Specifies additional connection parameters used when connecting to the database server.

Syntax

```
SQLCONNECT=parameter=value; ...
```

Remarks

This string is a list of parameter settings, of the form `parameter=value`, delimited by semicolons.

Connection parameters specified by the SQLCONNECT environment variable are not used if they have already been specified in the connection string.

Caution

Because the password is in plain text, putting it into the SQLCONNECT environment variable is a security risk.

As an alternative to the SQLCONNECT environment variable, you can put the connection string information into a file and then use @filename on the SQL Anywhere tool command line.

Note

This is only an option when using one of the SQL Anywhere tools.

Any connection parameter that has been specified in the SQLCONNECT environment variable will override any parameter that is contained in a DSN.

Example

The following is an example of putting the connection string information into a file and using the filename on the SQL Anywhere tool command line:

```
echo -c uid=dba;pwd=mypass > connect_info
dbping @connect_info
```

The following sequence will attempt to connect to the Sales database using user George's (not Henry's) password "mypass":

```
dbdsn -w mydsn -c uid=henry
set sqlconnect=uid=george
dbping -c dsn=mydsn;pwd=mypass;dbn=sales -d
```

Related Information

Alphabetical List of Connection Parameters [page 45]
UNIX and Linux Environment Variables [page 567]

1.5.1.19 SQLPATH Environment Variable

Specifies the location of SQL script and Help files.

Syntax

```
SQLPATH=path-list
```

Remarks

Interactive SQL searches the directories specified in SQLPATH for script files and Help files before searching the system path.

Related Information

UNIX and Linux Environment Variables [page 567]

1.5.1.20 SQLREMOTE Environment Variable

Specifies the directory that contains addresses for the SQL Remote FILE message link.

Syntax

```
SQLREMOTE=path
```

Remarks

You use the SET REMOTE FILE OPTION statement to specify the directory under which messages are stored. As an alternative, you can specify the SQLREMOTE environment variable.

Addresses for the FILE message link in SQL Remote are subdirectories of the SQLREMOTE environment variable. This environment variable should specify a shared directory.

On Windows operating systems, an alternative to setting the SQLREMOTE environment variable is to set the `HKEY_CURRENT_USER\Software\SAP\SQL Remote\FILE\Directory` registry entry to a string that indicates the full path of the subdirectory. Instead of HKEY_CURRENT_USER, you can also use the HKEY_LOCAL_MACHINE hive. HKEY_CURRENT_USER has precedence over HKEY_LOCAL_MACHINE.

The order of precedence from lowest to highest is:

- Registry setting (Windows only)
- SQLREMOTE environment variable
- SET REMOTE FILE OPTION statement

Related Information

UNIX and Linux Environment Variables [page 567]
The FILE Message System

1.5.1.21 SYBASE Environment Variable

Specifies the home directory for the installation of some SAP applications, including SAP Adaptive Server Enterprise, SAP Open Client, SAP Open Server, and utilities such as DSEdit.

Syntax

```
SYBASE=directory-name
```

Remarks

You only need to set this environment variable if you are using other SAP applications.

Related Information

UNIX and Linux Environment Variables [page 567]

1.5.1.22 TMP, TMPDIR, and TEMP Environment Variables

Specify the location of SQL Anywhere temporary files.

Syntax

```
TMP=path
```

```
TMPDIR=path
```

```
TEMP=path
```

Remarks

SQL Anywhere software may create temporary files for various operations, including a temporary file that is created when the database server starts and erased when the database server stops. As its name suggests, the temporary file is used to hold temporary information while the database server is running. The temporary file does not hold information that needs to be kept between sessions.

Temporary files are held in the directory specified by one of the TMP, TMPDIR, or TEMP environment variables. If more than one of these environment variables is specified, then the first of TMP, TMPDIR, and TEMP is used. You can also set the SATMP environment variable to specify the location of temporary files used by the database server and the SQL Anywhere command line utilities that require a temporary directory.

The location of the temporary file that is used by the database server can be specified when starting the database server using the -dt server option. If you do not specify a location for the temporary file when starting the database server, SQL Anywhere checks the following environment variables, in order: SATMP TMP TMPDIR TEMP. If an environment variable is not defined, SQL Anywhere places its temporary file in the current directory for Windows, and in the `/tmp` directory for Unix.

On Unix, the temporary directory is used to establish a shared memory connection.

Note

Using shared memory connections on Unix with older software

In SQL Anywhere version 9 and earlier, the environment variable ASTMP is equivalent to SATMP. If you are using shared memory to connect version 9 and version 12 software, you must set the SATMP and ASTMP environment variables to specify the location of the temporary file.

Related Information

UNIX and Linux Environment Variables [page 567]
-dt Database Server Option [page 416]
SATMP Environment Variable [page 578]

Troubleshooting: Database Server Startup [page 1023]

1.5.2 File Locations and Installation Settings

When you install SQL Anywhere, several directories are created.

Some of the files in these directories are essential, and others are not.

SQL Anywhere software, whether you receive it as a product or bundled as part of another product, is installed under a single installation directory. The SQLANY17 environment variable specifies the location of the installation directory.

SQL Anywhere installation directory

The SQL Anywhere installation directory itself holds several items, including the following:

Read Me First

A Read Me First file holds last minute information.

There are several directories under the installation directory:

Executable directories

There is a separate directory for each operating system platform, which holds configuration files and context-sensitive help files.

On Windows, these files are installed in the `bin32` or `bin64` directory. If you are using UNIX or Linux, they are installed in the `bin32` or `bin64` and `lib32` or `lib64` directories.

You only have the directories required for your operating system version.

java directory

JAR files are stored in this directory.

scripts directory

The scripts directory contains SQL scripts that are used by the database administration utilities and as examples.

SDK\Include directory

The `SDK\Include` directory contains header files for developing C/C++ applications for SQL Anywhere. On UNIX and Linux, this directory is called `include`.

UNIX and Linux File Locations

The language resources are installed in the `res` directory, and the shared objects are installed in the `lib32` or `lib64` directory.

In this section:

Related Information

Sourcing sa_config.sh, sample_config32.sh, and sample_config64.sh [UNIX/Linux] [page 568]
SQLANYSAMP17 Environment Variable [page 581]
SQLANY17 Environment Variable [page 580]

1.5.2.1 Samples Directory

When you install SQL Anywhere, you can choose the directory where the samples are installed. The installer creates the %SQLANYSAMP17% environment variable to identify this location.

Related Information

Sourcing sa_config.sh, sample_config32.sh, and sample_config64.sh [UNIX/Linux] [page 568]
SQLANYSAMP17 Environment Variable [page 581]
SQLANY17 Environment Variable [page 580]

1.5.2.2 How SQL Anywhere Locates Files

The client library and the database server need to locate files for two main purposes.

- DLLs and initialization files are required to run SQL Anywhere. If an incorrect DLL is located, there is the possibility of version mismatch errors.
- Some files are specified in SQL statements and need to be located at run time, such as INSTALL JAVA or LOAD TABLE.

Examples of SQL statements that use file names include the following:

INSTALL JAVA statement

The name of the file that holds Java classes.

LOAD TABLE and UNLOAD TABLE statements

The name of the file from which data should be loaded or to which the data should be unloaded.

CREATE DATABASE statement

A file name is needed for this statement and similar statements that can create files.

Sometimes SQL Anywhere uses a simple algorithm to locate files. In other cases, a more extensive search is performed.

In this section:

Simple File Searching [page 588]
When performing file searches using SQL, the file name is interpreted as relative to the current working directory of the database server.

Extensive File Searching on Windows, Linux, and UNIX [page 588]
SQL Anywhere software, including the database server and administration utilities, can perform a more extensive search for files such as executable programs, shared libraries, scripts, and profiles.

1.5.2.2.1 Simple File Searching

When performing file searches using SQL, the file name is interpreted as relative to the current working directory of the database server.

Also, when a database server is started and a database file name (DatabaseFile (DBF) parameter) is supplied, the path is interpreted as relative to the current working directory.

1.5.2.2.2 Extensive File Searching on Windows, Linux, and UNIX

SQL Anywhere software, including the database server and administration utilities, can perform a more extensive search for files such as executable programs, shared libraries, scripts, and profiles.

In these cases, SQL Anywhere software looks for files in the following order:

1. If an absolute path is specified, then the path is verified, and the search does not continue. On Windows, drive relative paths are not permitted (for example, C:file.txt or \file.txt) and the component requesting the search may fail the file reference.
2. On Windows only, the current module's directory is searched. This is the directory where the currently executing program executable file or library file (DLL) is located. For libraries, this is usually the Bin64 or Bin32 folder.
3. The current executable's directory is searched. This is the directory where the currently executing program executable file is located. It may be the same directory as in 2 above.
4. The SQL Anywhere installation path, specified by the *SQLANY*17 environment variable , and certain subdirectories like `bin64`, `bin32`, and `java` are searched. The subfolders searched depend on the file type. For example, JAR files are searched in the Java subdirectory only. For most software deployments, it is required that *SQLANY*17 be defined in the environment to permit the software to run successfully.

5. On UNIX/Linux systems, shared objects are searched using the library path environment variable:
   - LD_LIBRARY_PATH on Linux and Solaris
   - LD_LIBRARY_PATH and SHLIB_PATH on HP-UX
   - LIBPATH on IBM AIX
   - DYLD_LIBRARY_PATH on macOS
6. For some types of files (non-binaries), the location specified by the file path is searched. This search may be relative to the current directory of the requesting software. Typically, this search is used for user-specified file paths.
7. For some types of files (non-binaries) on UNIX/Linux systems, the product-specific subdirectory of the user's home directory is searched. This subdirectory is `$HOME/.sqlanywhere17`.
8. For some types of files (non-binaries) on Windows, a product-specific subdirectory in the AppData folder (%APPDATA%) is searched. This subdirectory is SQL Anywhere *17* .
9. For some types of files (non-binaries) on Windows, a product-specific subdirectory in the Common AppData folder (%ALLUSERSPROFILE%) is searched. This subdirectory is *SQL Anywhere 17*.
10. The folders specified by the PATH environment variable are searched last. Shared objects on UNIX/Linux systems are not searched using the PATH. You should exercise care when choosing what directories are listed in the PATH since they could be accessed by the software and applications.

1.5.2.3 Registry and INI Files

On Windows operating systems, SQL Anywhere uses several registry settings. On UNIX, these settings are stored in initialization files instead.

The software installation makes these settings for you, and in general operation you should not need to access the registry or initialization files. The settings are provided here for those people who make modifications to their operating environment.

The contents of `.ini` files used by SQL Anywhere can be encrypted using the File Hiding utility (dbfhide).

> ⚠ Caution
>
> You should not encrypt the system information file (named `.odbc.ini` by default) using the File Hiding utility (dbfhide) on Unix unless you are using SQL Anywhere data sources only.
>
> If you plan to use other data sources (for example, for MobiLink synchronization), then encrypting the contents of the system information file may prevent other drivers from functioning properly.

In this section:

Registry Settings on Installation [page 592]
On Windows, the installation program creates settings in the `HKEY_LOCAL_MACHINE\Software\SAP` registry key.

1.5.2.3.1 Current User and Local Machine Settings

Some operating systems hold two levels of system settings: current user settings, and local machine settings.

Some settings are specific to an individual user and are used only when that user is logged on; these settings are called **current user** settings. Some settings are global to the computer, and are available to all users; these are called **local machine** settings. You must have administrator permissions on your computer to change local machine settings.

SQL Anywhere respects both current user and local machine settings. On Windows, for example, these are held in the `HKEY_CURRENT_USER` and `HKEY_LOCAL_MACHINE` registry keys, respectively.

Current User Takes Precedence

If a setting is made in both the current user and local machine registries, the current user setting takes precedence over the local machine setting.

When Local Machine Settings are Needed

If you are running a SQL Anywhere program as a **service**, you should ensure that the settings are made at the local machine level.

Services can continue to run under a special account when you log off a computer as long as you do not shut the computer down entirely. They can be made independent of individual accounts, and therefore need access to local machine settings.

In addition to SQL Anywhere programs, some web servers run as services. You must set local machine settings for Apache or Microsoft IIS to work with such a web server.

In general, the use of local machine settings is recommended.

1.5.2.3.2 Hiding the Contents of an `.ini` File

Use encryption to protect passwords or other information in an `.ini` file with the File Hiding (dbfhide) utility.

Context

SQL Anywhere expects an `.ini` file to have a particular name. When you encrypt a file whose name is important (such as `saldap.ini`), you must save a copy of the original unencrypted file under a different name. If you do not keep a copy of the original file, then you cannot modify the contents of the file once it has been encrypted.

> ⚠ Caution
>
> You should not encrypt the system information file (named `.odbc.ini` by default) using the File Hiding utility (dbfhide) on UNIX or Linux unless you will be using SQL Anywhere data sources only.
>
> If you plan to use other data sources (for example, for MobiLink synchronization), then encrypting the contents of the system information file may prevent other drivers from functioning properly.

Procedure

1. Save the file with a different name.

   ```
   rename saldap.ini saldap.ini.org
   ```

2. Encrypt the file with the File Hiding utility, giving the encrypted file the required file name.

   ```
   dbfhide saldap.ini.org saldap.ini
   ```

3. Protect the `saldap.ini.org` file using file system or operating system protection, or store the file in a secure location.

   To make a change to the `saldap.ini` file, edit the `saldap.ini.org` file and repeat step 2.

Results

The contents of the `.ini` file are hidden.

Related Information

File Hiding Utility (dbfhide) [page 1160]

1.5.2.3.3 Registry Structure

On Windows, you can access the registry directly with the registry editor.

The SQL Anywhere registry entries are held in either the `HKEY_CURRENT_USER` or `HKEY_LOCAL_MACHINE` keys, in the following location:

```
Software\SAP\SQL Anywhere\17.0
```

> ⚠ Caution
>
> Modify your registry at your own risk. It is recommended that you back up your system before modifying the registry.

1.5.2.3.4 Registry Settings on Installation

On Windows, the installation program creates settings in the `HKEY_LOCAL_MACHINE\Software\SAP` registry key.

The following list describes some of these registry settings:

SQL Anywhere\17.0\Location

This entry holds the installation directory location for the SQL Anywhere software. For example:

```
Location "C:\Program Files\SQL Anywhere 17"
```

SQL Anywhere\17.0\Samples Location

This entry holds the installation directory location for sample programs. For example:

```
Samples Location "C:\Users\Public\Documents\SQL Anywhere
         17\Samples"
```

SQL Anywhere\17.0\Online Resources

This entry holds the location for the Online Resources documentation. For example:

```
Online Resources "C:\Program Files\SQL Anywhere 17\support
\OnlineResources.html"
```

SQL Anywhere\17.0\Language

This entry holds a two-letter code indicating the current language for messages and errors. For example:

```
Language "EN"
```

The language is set based on the language selection specified during installation.

> ⚠ Caution
>
> To improve robustness across power failures on systems using certain Intel storage drivers, specific registry entries must be set. Failure to set this parameter can result in lost data and corrupted databases in the event of a power failure.

Related Information

Locale Language [page 614]
Improving Robustness on Intel Storage Drivers

1.5.3 Time Zone Management

A database can use a simulated time zone rather than the time zone of the host computer.

By default, the database server uses the time zone of the computer on which the database server is running on. However, you can configure a database to run as though it is located in a different time zone. When the database runs in this simulated time zone, time calculations and functions use the simulated time zone. Only the scheduled events, transaction logs, and auditing records continue to use the time zone of the database server, not the simulated time zone.

This feature allows you to run different databases from the same database server and have each database appear to be running from a different time zone. For example, a company could host three databases on the same database server in Germany for customers in London, Rome, and Shanghai, and have the time zone of each database reflect that of the customer city.

This feature also allows you to run databases in different locations and have them all appear to be running in the same time zone. For example, you could have databases located in Alaska, New York, and Los Angeles, but have them all appear to running from Los Angeles. To determine the time zone that a database is running in, you can query the timezone or CurrentTimezoneOffset database property or the SYSTIMEZONE system view. The TimeZoneAdjustment database server property can be used to calculate the time zone of the database server.

Using a simulated time zone affects calculations using the CURRENT TIME special value or the NOW or GETDATE functions, but does not affect the value of the CURRENT UTC TIME or CURRENT SERVER TIME special values. Queries that result in a time, date, or timestamp (with the exception of values returned from tables) are returned using the database's time zone, if the database has the time_zone option set.

> **i Note**
>
> If you run a query against another database that returns a time, date, or timestamp, then that value is returned using the time zone of the database that you are currently connected to, *not* the time zone of the database you are running the query against.

In this section:

Creating Simulated Time Zones (SQL Central) [page 594]
Use SQL Central to create a time zone and configure the database to run in this simulated time zone.

Creating Simulated Time Zones (SQL) [page 594]
Use Interactive SQL to create a time zone and configure the database to run in this simulated time zone.

1.5.3.1 Creating Simulated Time Zones (SQL Central)

Use SQL Central to create a time zone and configure the database to run in this simulated time zone.

Prerequisites

You must have the MANAGE TIME ZONE system privilege.

Context

By default, the database server uses the time zone of the computer on which the database server is running. This task configures the database to run as though it is located in a different time zone. When the database runs in this simulated time zone, time calculations and functions use the simulated time zone. Only the transaction log and auditing records continue to use the time zone of the database server, not the simulated time zone.

Procedure

1. In the SQL Central, click *Time Zones* in the left pane.
2. In the right pane, right-click and click *New* > *Time Zone*.

Results

The new time zone is created, and by default it is set as the current time zone for the database.

1.5.3.2 Creating Simulated Time Zones (SQL)

Use Interactive SQL to create a time zone and configure the database to run in this simulated time zone.

Prerequisites

You must have the MANAGE TIME ZONE system privilege.

Context

By default, the database server uses the time zone of the computer on which the database server is running. This task configures the database to run as though it is located in a different time zone. When the database runs in this simulated time zone, time calculations and functions use the simulated time zone. Only the transaction log and auditing records continue to use the time zone of the database server, not the simulated time zone.

Procedure

1. The following steps create a simulated time zone based on Australian Eastern Standard Time, and configures your database to use this time zone. In Interactive SQL, connect to your database.
2. Execute the following statement to create a time zone called NewSouthWales. The time zone is 10 hours ahead of UTC. Daylight savings time begins on the first Sunday of October at 2:00 AM, and ends on the first Sunday of April at 2:00 AM.

   ```
   CREATE TIME ZONE NewSouthWales OFFSET '10:00'
   STARTING 'Oct/Sun>=1' AT '2:00'
   ENDING 'Apr/Sun>=1' AT '2:00';
   ```

3. Execute the following statement to configure your database to use the simulated time zone NewSouthWales.

   ```
   SET OPTION PUBLIC.time_zone='NewSouthWales';
   ```

Results

The new time zone is created, and by default it is set as the current time zone for the database.

1.5.4 International Languages and Character Sets

Internationalization refers to the ability of software to handle a variety of languages and their appropriate characters sets, independently of the language in which the software is running, or the operating system on which the software is running.

The following features discuss the most commonly requested and used capabilities.

Unicode support

Unicode is supported as follows:

- Client support for UTF-16 in SQL Anywhere client libraries for ODBC, OLE DB, ADO.NET, and JDBC
- NCHAR data types for storing Unicode character data in UTF-8
- CHAR data types can use UTF-8 encoding

Code pages and character sets

The database server and related tools support Windows (ANSI/ISO), UTF-8, and Unix code pages and character sets.

Collations

Two collation algorithms are supported: the SQL Anywhere Collation Algorithm (SACA), and the Unicode Collation Algorithm (UCA) using International Components for Unicode (ICU).

SACA provides fast, compact, and reasonable sorting at the expense of linguistic correctness. UCA provides linguistic correctness, but with a small expense in storage requirements and execution time.

For advanced ordering and comparison capabilities, the SORTKEY and COMPARE functions are also provided. These functions provide advanced linguistic sorting capabilities, like the ordering found in a dictionary or telephone book. Where appropriate, case-insensitive and accent-insensitive ordering and comparisons are provided.

Design features allowing for automatic use of SORTKEY-based ordering on character columns are also included. The sort_collation database option specifies the sort ordering to be used when an ORDER BY is specified for a character column. Computed columns may also be used to store sort keys for character columns so that they do not need to be computed each time that an ORDER BY is specified.

Character set conversion

Data is converted between the character set encoding on your server and client systems. The integrity of your data is maintained even in mixed character set environments.

Identifiers

The use of identifiers containing most single-byte and multibyte characters are supported without requiring quotes. Exceptions are generally limited to spaces and punctuation symbols.

Currency

Currency symbols, including the euro symbol, are supported for ordering. No currency formatting support is provided.

Date and time formats

The Gregorian calendar is supported, and a variety of formats for date and time strings are provided. Custom formatting can be done using the date_format, time_format, and timestamp_format database options. The date_format and timestamp_format options default to an ISO-compatible format for the date, YYYY-MM-DD. The CONVERT function, which provides output formatting of dates and times into a variety of popular formats, is provided.

In this section:

Localized Versions of the Software [page 597]
Localization refers to the linguistic and cultural adaptation of a product to a target locale, which is usually a combination of language and country/region.

Character Sets [page 601]
A **character set** is a set of symbols, including letters, digits, spaces, and other symbols.

Character Set Conversion [page 606]
Character set conversion can be performed between character sets that represent the same characters, but at different positions in the character set or code page.

Locales [page 613]
Both the database server and the client library recognize their language and character set environment using a **locale definition**.

Collations [page 618]
A collation describes how to sort and compare characters from a particular character set or encoding.

Recommended Character Sets and Collations [page 634]
While the names of hundreds of character sets, code pages, encodings, and collations are recognized, there are several that are recommended for use with Windows and UNIX/Linux platforms, depending on the language in use.

Turkish Character Sets and Collations [page 637]
The Turkish language has two forms of what appears to be the letter *I*.

Related Information

What Is ICU and When Is It Needed? [page 599]
Collation Considerations [page 627]
SORTKEY Function [String]
COMPARE Function [String]
sort_collation Option [page 821]
date_format Option [page 716]
time_format Option [page 847]
timestamp_format Option [page 850]
CONVERT Function [Data Type Conversion]

1.5.4.1 Localized Versions of the Software

Localization refers to the linguistic and cultural adaptation of a product to a target locale, which is usually a combination of language and country/region.

Localization affects many components, including packaging, installation, documentation, software user interface, and error/warning/information messages.

SQL Anywhere software is localized to five languages:

- English
- French
- German
- Japanese
- Simplified Chinese

Language choice is determined at installation.

Localized versions of the documentation are available in English, German, Japanese, and Simplified Chinese.

On Windows, *Start* menu items allow the software to be reconfigured between the installed language and English. The Language Selection utility (dblang) allows the software to be reconfigured to any of the available languages, including the additional deployment languages.

On all Windows and Linux platforms, the software is available in English, French, German, Japanese, and Simplified Chinese.

On all other Unix platforms and macOS, the software is only available in English.

In this section:

Related Information

Language Selection Utility (dblang) [page 1186]

1.5.4.1.1 Full Software and Documentation Localization

The software and documentation is available in several languages, suitable for development, deployment, and administration.

- English
- French
- German
- Japanese
- Simplified Chinese

For English, German, Japanese, and Simplified Chinese, all components are localized, including:

- Installer
- Documentation and context-sensitive help
- Software

For French, the installer, software, and context-sensitive help are localized.

1.5.4.1.2 Deployment Software Localization on Microsoft Windows

In addition to the five main languages (English, French, German, Japanese, and Simplified Chinese), there are deployment software resources available for several other languages.

- Italian
- Korean
- Lithuanian
- Polish
- Portuguese (Brazilian)
- Russian
- Spanish
- Traditional Chinese
- Ukrainian

Deployment localization applies to a subset of software components typically deployed to end users. Packaging, documentation, administration, development, and installation software are not localized. Localized software components include:

- Database servers and client libraries
- MobiLink server and client
- SQL Remote client
- Command-line tools, such as dbinit and dbunload

1.5.4.1.3 What Is ICU and When Is It Needed?

ICU, or International Components for Unicode, is an open source library developed and maintained by IBM. ICU facilitates software internationalization by providing Unicode support.

Certain character set conversions and collation operations are implemented using ICU.

When Is ICU Needed on the Database Server?

Ideally, ICU should always be available for use by the database server. The following table specifies when and why ICU is needed:

ICU is needed when...	Notes
Unicode Collation Algorithm (UCA) is used as the collation for the NCHAR or CHAR character set.	UCA requires ICU.
The database character set is not UTF-8, but is a multi-byte character set.	For password conversion from the database character set to UTF-8 (database passwords are stored in UTF-8, internally).

ICU is needed when...	Notes
The client and database character sets are different, and when either of them is multi-byte (including UTF-8). This includes Unicode ODBC, OLE DB, ADO.NET, and SQL Anywhere JDBC applications, regardless of the database character set where at least one of these clients does not have ICU.	Proper conversion to and from a multi-byte character set requires ICU.
The database character set is not UTF-8 and conversion between CHAR and NCHAR values is required.	The database server requires ICU to convert UTF-8 to another character set.
An Embedded SQL client uses an NCHAR character set other than UTF-8.	The database server requires ICU to convert UTF-8 to another character set. The default Embedded SQL client NCHAR character set is the same as the initial client CHAR character set. This can be changed using the db_change_nchar_charset function.
The CSCONVERT or SORTKEY functions are used. The CSCONVERT function is called to convert between character sets that conform to the requirements of the third point above.	Character set conversion o and from a multi-byte character set requires ICU. Sortkey generation for many sortkey labels requires UCA, which, in turn, requires ICU.

When Can I Get Correct Character Set Conversion on the Database Server Without ICU?

You can get correct character set conversion without ICU when both the database character set and client character set are single-byte and `sqlany.cvf` is available (all platforms), or if the operating system supports the conversion (Windows only). This is because single-byte to single-byte conversions can be processed without ICU if the `sqlany.cvf` file is available, or the host operating system has the appropriate converters installed.

When Is ICU Needed on the Client?

For Unicode client applications, you are likely to get better combined client and database server performance when all clients have ICU installed, regardless of the database character set. This is because some of the required conversion activity may be offloaded from the database server to the client, and because fewer conversions are required.

Also, if you are using ODBC on Windows platforms, you must have ICU installed on the client, even for ANSI applications. This is because the driver manager converts ANSI ODBC calls to Unicode ODBC calls.

Related Information

db_change_nchar_charset Function
CSCONVERT Function [String]
SORTKEY Function [String]

1.5.4.2 Character Sets

A **character set** is a set of symbols, including letters, digits, spaces, and other symbols.

Each piece of software works with a character set. An example of a character set is ISO-8859-1, also known as Latin1.

To properly represent these characters internally, each piece of software employs an **encoding**, also known as **character encoding**. An encoding is a method by which each character is mapped onto one or more bytes of information, and is presented as a hexadecimal number. An example of an encoding is UTF-8.

Sometimes the terms character set and encoding are used interchangeably, since the two aspects are so closely related.

A **code page** is one form of encoding. A code page is a mapping of characters to numeric representations, typically an integer between 0 and 255. An example of a code page is Windows code page 1252.

In this documentation, the terms encoding, character encoding, character set encoding, and code page are synonymous.

Database servers, which sort characters (for example, listing names alphabetically), use a **collation**. A collation is a combination of a character encoding (a map between characters and their representation) and a **sort order** for the characters. There may be more than one sort order for each character set; for example, a case sensitive order and a case insensitive order, or two languages may sort the same characters in a different order.

Characters are printed or displayed on a screen using a **font**, which is a mapping between characters in the character set and their appearance. Fonts are handled by the operating system.

Operating systems also use a **keyboard mapping** to map keys or key combinations on the keyboard to characters in the character set.

In this section:

1.5.4.2.1 Language Issues in Client/Server Computing

Database users working at client applications may see or access strings from several sources.

Data in the database

Strings and other text data are stored in the database. The database server processes these strings when responding to requests. For example, the database server may be asked to supply all the last names beginning with a letter ordered less than N in a table. This request requires string comparisons to be performed, and assumes a character set ordering.

Database server software messages

Applications can cause database errors to be generated. For example, an application may submit a query that references a column that does not exist. In this case, the database server returns a warning or error message. This message is held in a **language resource library**, which is a DLL or shared library used by the database server.

Client application

The client application interface displays text, and internally the client application may process text.

Client software messages

The client library uses the same language library as the database server to provide messages to the client application.

Operating systems

The client and server operating systems may provide messages or process text.

For a satisfactory working environment, all these sources of text must work together. Loosely speaking, they must all be working in the user's language and/or character set.

1.5.4.2.2 Single-byte Character Sets

Many languages have few enough characters to be represented in a single-byte character set.

In such a character set, each character is represented by a single byte: a two-digit hexadecimal number.

At most, 256 characters can be represented in a single byte. No single-byte character set can hold all the characters used internationally, including accented characters. This problem was addressed by the development of a set of code pages, each of which describes a set of characters appropriate for one or more national languages. For example, code page 1253 contains the Greek character set, and code page 1252 contains Western European languages. There are many code pages, and many names for code pages. The above examples are code pages for Windows.

Upper and Lower Pages

With few exceptions, characters 0 to 127 are the same for all the code pages. The mapping for this range of characters is called the **ASCII** character set. It includes the English language alphabet in upper and lowercase, and common punctuation symbols and the digits. This range is often called the **seven-bit** range (because only

seven bits are needed to represent the numbers up to 127) or the **lower** page. The characters from 128 to 255 are called **extended characters**, or **upper** code page characters, and vary from one code page to another.

Problems with code page compatibility are rare if the only characters used are from the English alphabet, as these are represented in the ASCII portion of each code page (0 to 127). However, if other characters are used, as is generally the case in any non-English environment, there can be problems if the database and the application use different code pages.

For example, suppose a database using the UTF-8 character set loads a table from a file containing cp1252 data, and the encoding is not specified as cp1252 on the LOAD TABLE statement. Because the encoding is not specified, the data is assumed to be encoded in UTF-8, so no character conversion takes place; the cp1252 encoding is stored directly in the database. Characters such as the euro symbol, represented in cp1252 as hex 80, are not converted into UTF-8. The euro symbol in UTF-8 is represented by the three-byte sequence E2 82 AC, but, in this case, is stored in the database as 80. Subsequently, when an application requests data, the database server attempts to convert the data from UTF-8 to the client character set. The conversion produces corrupted characters.

1.5.4.2.3 Multibyte Character Sets

Some languages have many more than 256 characters, which cannot be represented using a single byte, so a multibyte encoding is used.

In addition, some character sets use the much larger number of characters available in a multibyte representation to represent characters from many languages in a single, more comprehensive, character set. An example of this is UTF-8.

Multibyte character sets may be of **variable width** whereby some characters are single-byte characters; others are double-byte, and so on.

Example

As an example, characters in code page 932 (Japanese) are either one or two bytes in length. If the value of the first byte, also called the **lead byte**, is in the range of hexadecimal values from \x81 to \x9F or from \xE0 to \xFC (decimal values 129-159 or 224-252), the character is a two-byte character and the subsequent byte, also called a **follow byte**, completes the character. A follow byte is any byte other than the first byte.

If the first byte is outside the lead byte range, the character is a single-byte character and the next byte is the first byte of the following character.

Related Information

SQL Anywhere Collation Algorithm (SACA) [page 621]

1.5.4.2.4 ANSI and OEM Code Pages in Microsoft Windows

For Windows users, there are two code pages in use: ANSI code pages and OEM code pages.

Applications using the Windows graphical user interface use the Windows code page. Windows code pages are compatible with ISO character sets, and also with ANSI character sets. They are often referred to as **ANSI code pages**.

Character-mode applications (those using a command prompt window) in Windows use code pages that were used in DOS. These are called **OEM code pages** (Original Equipment Manufacturer) for historical reasons.

Collations based on both OEM and ANSI code pages are supported. The OEM collations are provided for compatibility, but they should not be used for new databases.

Related Information

Alternate Collations [page 624]

1.5.4.2.5 Character Sets in a Database

A database can use one or two character sets (encodings) for storing character data.

The CHAR data types, including CHAR, VARCHAR, and LONG VARCHAR, use a single-byte or multibyte character set. UTF-8 can be used. The NCHAR data types, including NCHAR, NVARCHAR, and LONG NVARCHAR, use UTF-8.

SQL statements like LOAD TABLE and functions like CSCONVERT, TO_CHAR, and TO_NCHAR take **db_charset** and **nchar_charset** as parameters to refer to the database CHAR character set and to the database NCHAR character set, respectively.

Related Information

Character Data Types
LOAD TABLE Statement
CSCONVERT Function [String]
TO_CHAR Function [String]
TO_NCHAR Function [String]

1.5.4.2.6 Supported Character Sets

A growing list of hundreds of character sets and labels is supported.

Character set encodings are known by a wide variety of names or labels. To view the list of character sets supported, run the following command:

```
dbinit -le
```

Each line of output lists the most common labels for a given character set encoding, in comma separated form. The first label in each line of output is the preferred name for the character set encoding. The others are the labels used by different authorities, organizations, or standards. These are IANA (Internet Assigned Numbers Authority), MIME (Multipurpose Internet Mail Extensions), ICU (International Components for Unicode), Java, and ASE (Adaptive Server Enterprise).

If you do not find the character set you are looking for, you can also run the following command to see a longer list that includes labels that are less common:

```
dbinit -le+
```

When a character set encoding label is specified, the database server searches for the label in the set of labels known to it. Different authorities sometimes use the same label for different character sets. The database server does its best to resolve ambiguities by context. For example, a JDBC application that references a character set by an ambiguous label resolves to a Java standard label. It is recommended that the SQL Anywhere label always be used to avoid any ambiguities.

In addition to the character set encoding labels returned by the dbinit -le option, you can also use the following character set aliases:

os_charset

Alias for the character set used by the operating system hosting the database server.

char_charset

Alias for the CHAR character set used by the database.

nchar_charset

Alias for the NCHAR character set used by the database.

An easy way to determine if a certain character set or label is supported is to test it using the CSCONVERT function.

Related Information

CSCONVERT Function [String]

1.5.4.2.7 Character Set Questions and Answers

There are several areas of the documentation you can use to find answers to questions regarding character sets.

To answer the question...	Consider searching and reading about...
How do I decide which collation to use for my database?	Collations
How are characters represented in software, and in SQL Anywhere in particular?	Character sets
What collations does SQL Anywhere provide?	Collation considerations
What character set encodings does SQL Anywhere support?	Supported character sets
I have a different character set on client computers from that in use in the database. How can I get characters to be exchanged properly between client and server?	Character set conversion
What character sets can I use for connection strings?	Connection strings and character sets
How do I change the collation sequence of an existing database?	Changing the database collation

Related Information

Collations [page 618]
Collation Considerations [page 627]
Supported Character Sets [page 605]
Character Set Conversion [page 606]
Connection Strings and Character Sets [page 607]
Changing the Database Collation [page 632]

1.5.4.3 Character Set Conversion

Character set conversion can be performed between character sets that represent the same characters, but at different positions in the character set or code page.

There needs to be a degree of compatibility between the character sets for this to be possible. For example, character set conversion is possible between EUC-JIS and cp932 character sets, but not between EUC-JIS and cp1252.

Character set conversion is implemented using the International Components for Unicode (ICU) open source library, developed and maintained by IBM.

In this section:

Connection Strings and Character Sets [page 607]
If all of your clients do not use the same character sets, connection strings may be a challenge during character set conversion.

SQL Statements and Character Sets [page 608]
Character set conversion causes all SQL statements to be converted to the database character set before parsing and execution.

Troubleshooting Unexpected Symbols When Viewing Data [page 608]
When selecting and viewing data using a client application such as Interactive SQL, unexpected symbols including squares, arrows, and question marks, may appear as characters in the data.

International Aspects of Case Sensitivity [page 609]
Identifiers are always case preserving and case insensitive.

Character Set Conversion and Client APIs [page 609]
When working in a multi-character set environment, character set conversion issues can occur and it can be difficult to determine where the conversion issue occurred.

Related Information

Data Type Comparisons

1.5.4.3.1 Connection Strings and Character Sets

If all of your clients do not use the same character sets, connection strings may be a challenge during character set conversion.

This is because the connection string is parsed by the client library to locate or start a database server. However, this parsing is done with no knowledge of the character set or language in use by the database server.

The interface library parses the connection string as follows:

1. The connection string is broken down into its `keyword=value` pairs. This can be done independently of the character set, as long as you do not use curly braces {} around CommLinks (LINKS) connection parameters. Instead, use the recommended parentheses (). Curly braces are valid **follow bytes** (bytes other than the first byte) in some multibyte character sets.
2. The database server is located. There is no character set conversion performed on the server name. If the client character set and the database server character set are different, using extended characters in the server name can cause the server to not be found.
   If your clients and servers are running on different operating systems or locales, you should use 7-bit ASCII characters in the database server name.
3. The DatabaseName (DBN) or DatabaseFile (DBF) connection parameters are converted from client character set to the database server character set.
4. Once the database is located, the remaining connection parameters are converted to the database's character set.

1.5.4.3.2 SQL Statements and Character Sets

Character set conversion causes all SQL statements to be converted to the database character set before parsing and execution.

A side-effect of this conversion is that any characters in the SQL statement that cannot be converted to the database character set are converted to a substitution character. A SQL statement with an arbitrary Unicode character can be executed in one of the following ways:

- Use the UNISTR function to specify the Unicode character values
- Use a host variable to specify the Unicode character values
- Use UTF-8 as the database character set

If you select UTF8BIN as the char collation, the database character set is UTF-8. If you select UCA as the CHAR collation, you can choose UTF-8 as the encoding.

The Unicode Collation Algorithm (UCA) provides advanced comparison, ordering, and case conversion, but it can affect performance. Although UTF8BIN is space-efficient and fast, the sort order and comparison is binary. Specify the char collation as UTF8BIN if you require Unicode characters in your SQL statements, but do not need the full power of UCA for sorting and comparison. Use UCA only when necessary, by using the SORTKEY and COMPARE functions.

Related Information

Unicode Collation Algorithm (UCA) [page 622]
SQL Anywhere Collation Algorithm (SACA) [page 621]
Connection Parameters and Connection Strings [page 35]
SORTKEY Function [String]
COMPARE Function [String]

1.5.4.3.3 Troubleshooting Unexpected Symbols When Viewing Data

When selecting and viewing data using a client application such as Interactive SQL, unexpected symbols including squares, arrows, and question marks, may appear as characters in the data.

The most common reason for the appearance of unexpected symbols is that the font used to display the data does not support the characters. You can resolve this problem by changing to a Unicode font. If it is not possible to change the font for the client application, you can change the operating system default font instead.

Unexpected symbols might also appear in the client application if there is a problem with the underlying data that is stored in the database. For example, if character set conversion was required when the data was inserted into the database, and some characters in the original character set did not have an equivalent character in the database character set, then substitution characters were inserted instead.

For example, in Windows, the standard English font Tahoma does not support Japanese characters. If your database character set is cp932 and the database contains Japanese data, when you query the database,

characters in the result set are displayed as small boxes. In Interactive SQL, you can change the font used in the *Results* pane by clicking *Tools* > *Options* > *SQL Anywhere* > *UltraLite* > *Results* > *Font*, and specifying a Unicode font such as Arial Unicode MS, or Lucida Sans Unicode. Unicode fonts are recommended as they are capable of displaying characters from many languages.

If your client application does not allow you to change font settings, it is likely using your default operating system font. In this case, consult your operating system documentation for information about how to change the default system font to a Unicode font.

Related Information

Lossy Conversion and Substitution Characters
Interactive SQL [page 1029]

1.5.4.3.4 International Aspects of Case Sensitivity

Identifiers are always case preserving and case insensitive.

The names are stored in the case in which they are created, but any access to the identifiers is done in a case insensitive manner.

For example, the names of the system views are stored in uppercase (SYSDOMAIN, SYSTAB, and so on), but access is case insensitive, so that the two following statements are equivalent:

```
SELECT * FROM systab;
SELECT * FROM SYSTAB;
```

The equivalence of upper and lowercase characters is defined in the collation. There are some collations where particular care is required when assuming case insensitivity of identifiers. For example, Turkish collations have a case-conversion behavior that can cause unexpected and subtle errors. The most common error is that a system object containing a letter *I* or *i* is not found.

Related Information

Turkish Character Sets and Collations [page 637]

1.5.4.3.5 Character Set Conversion and Client APIs

When working in a multi-character set environment, character set conversion issues can occur and it can be difficult to determine where the conversion issue occurred.

When encountering character set conversion issues for client APIs, examine the database and connection options and properties that control character set conversion.

There are two categories into which conversion issues can be placed. The first involves sending data in the wrong format to the client API. Although this cannot happen with Unicode APIs, it is possible with all other client APIs, and results in garbled data.

The second category of issue involves a character that does not have an equivalent in the final character set, or in one of the intermediate character sets. In this case, a substitution character is used. This is called lossy conversion and can happen with any client API. You can avoid lossy conversions by configuring the database to use UTF-8 for the database character set.

Option and property settings that impact character set conversion

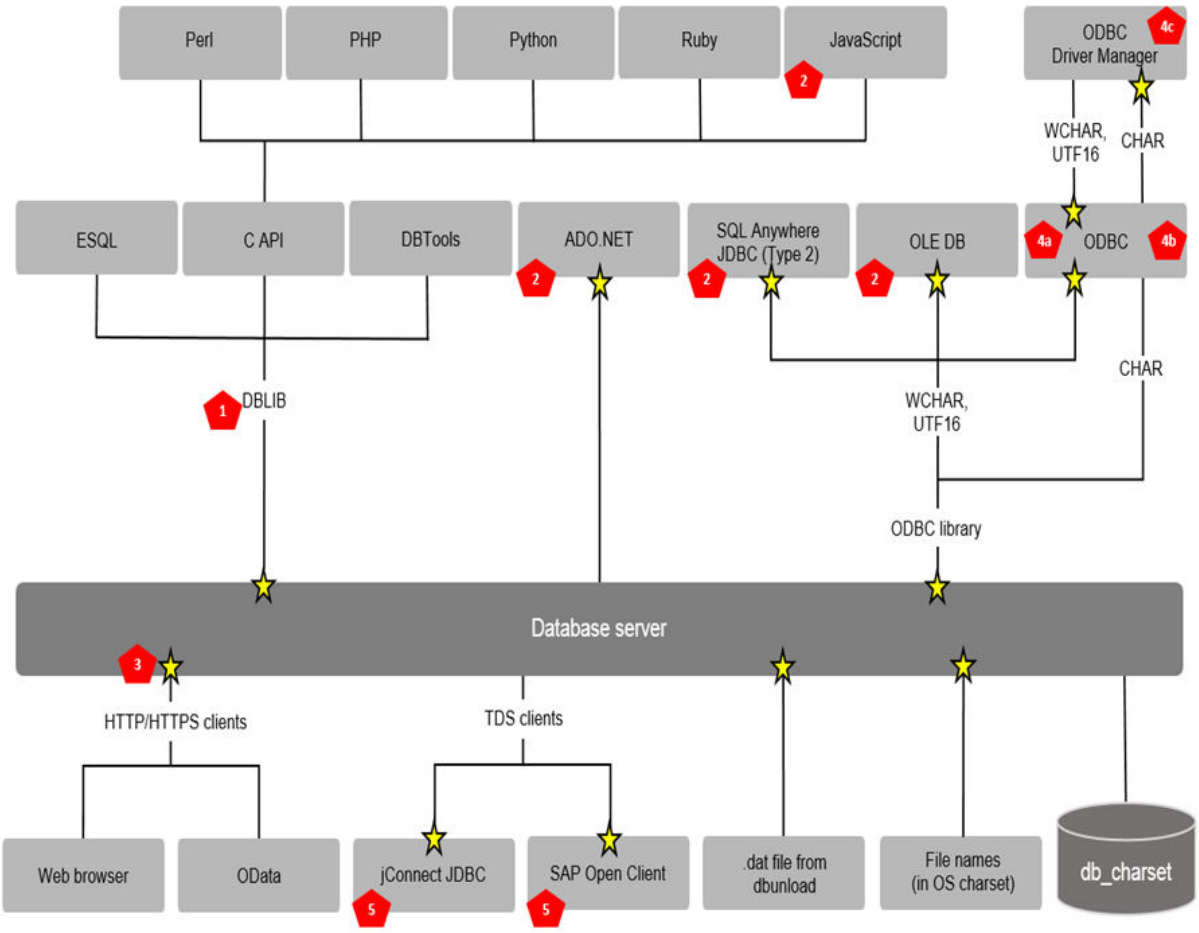
The database options and database and connection properties that are in effect for a connection are typically available in the full connection string. However, you can also query for the settings using system functions such as PROPERTY, DB_PROPERTY, and CONNECTION_PROPERTY. For example:

Query	Description
SELECT PROPERTY('CharSet');	Returns the database server's operating system character set.
SELECT DB_PROPERTY('CharSet');	Returns the database CHAR character set.
SELECT DB_PROPERTY('NcharCharSet');	Returns the database NCHAR character set.
SELECT DB_PROPERTY('MultibyteCharSet');	Returns whether CHAR data uses a multibyte character set (On=yes, Off=no).
SELECT CONNECTION_PROPERTY('CharSet');	Returns the client CHAR character set.
SELECT CONNECTION_PROPERTY('NcharCharSet');	Returns the client NCHAR character set.
SELECT CONNECTION_PROPERTY('on_charset_conversion_failure');	Returns the value of the on_charset_conversion_warning option.

Client API and points of character set conversion

The following diagram shows the possible locations of character set conversion when client APIs interact with the database server.

The yellow stars in the diagram indicate a location where character set conversion can take place. The numbers in the diagram correspond to additional notes located after the diagram.



1 - DBLIB

DBLIB CHAR and NCHAR character sets default to the client operating system character set. The CHAR character set is used for all string data except BINARY and NCHAR. The NCHAR character set is used for NCHAR typed host variables and host parameters (these are not available from the DBTools or C API interfaces). The database server performs all character set conversion.

You can set the CHAR character set and the NCHAR character set using the CharSet connection parameter and the db_change_nchar_charset function, respectively.

2 - Unicode APIs

With Unicode APIs, a driver uses the Unicode UTF-16 encoding for character data, and converts Unicode data to/from the database character set. The driver converts Unicode host parameters to the NCHAR character set (UTF-8) before sending them to the database server. Result set columns that are described as NCHAR are fetched in the NCHAR character set (UTF-8) and converted to Unicode after they are received. Setting the CharSet connection parameter is not recommended because it can result in a lossy conversion.

3 - HTTP/HTTPS clients

HTTP services (SQL Anywhere as the server): For HTTP services, there are two types of requests: URL-encoded or multipart/form-data requests.

URL-encoded

Requests take the form of application/x-www-form-urlencoded where variables are passed as key/value pairs. The database server automatically decodes %-encoded data (UTF-8 or database character set) and translates the key/value pairs into the database character set. Processed values can be extracted using the HTTP_VARIABLE function where the @BINARY or @TRANSPORT attributes can be used to return a value that is %-decoded but not character set translated, or the raw HTTP (transport) value, respectively.

Multipart/form-data

Requests are considered to be binary (using the attributes @BINARY or @TRANSPORT would return identical values).

Character set conversion of an HTTP service response is controlled by the CharsetConversion and AcceptCharset HTTP options, which you can set using the sa_set_http_option system procedure.

Other settings that can impact character set conversion for requests and responses are the Accept-Charset header, which is used to specify the preferred character sets, and the Content-Type header, which is used to identify the character set of the content.

HTTP stored procedures (SQL Anywhere as the client): For HTTP stored procedures, CHAR data is sent in the database character set. If any parameter is an NCHAR type, then all data is sent as UTF-8 (all CHAR parameters are translated to UTF-8). The request sends an Accept-Charset HTTP header identifying the database character set as the preferred character set. UTF-8 is always included in the list of preferred character sets. If the response specifies a character set in its Content-Type header that is not the database character set, then the client translates the response into the database character set.

4a - ODBC with Unicode entry points

If an ODBC application uses the Unicode entry points, it is considered a Unicode client API. WCHAR data is handled in the same way it is handled for Unicode APIs. If an ODBC application uses the Unicode entry points and CHAR (ANSI) data, the data is assumed to be in the database character set. Mixing CHAR and WCHAR data in the same application is not recommended.

4b - ODBC with ANSI entry points

If an ODBC application uses the ANSI entry points, it is considered an ANSI client API. The CHAR character set defaults to the client operating system character set. The CHAR character set can be changed using the CharSet connection parameter.

Any conversion of ANSI data is done by the database server. Fetching into WCHAR host variables can result in a lossy conversion for CHAR data on the database server. Mixing CHAR and WCHAR data in an application is not recommended.

4c - ODBC driver managers

Some ODBC driver managers convert all CHAR data to WCHAR data and then call the Unicode entry points.

5 - TDS clients

TDS clients such as SAP Open Client and jConnect negotiate with the database server at connect time and prefer that character set conversion take place on the client. During the negotiation, the database server is instructed not to perform character set conversion.

Related Information

Lossy Conversion and Substitution Characters

Data Type Conversions
Lossy Conversion and Substitution Characters
NCHAR to CHAR Conversions
SQL Anywhere C API Support
Perl DBI Support
Python and Database Access
SQL Anywhere PHP Extension
Ruby Programming
SAP Open Client Support
The Database Server as an HTTP Web Server
OLE DB
SQL Anywhere .NET Data Provider
ODBC Support
JDBC Support
db_change_nchar_charset Function
sa_set_http_option System Procedure
CharSet (CS) Connection Parameter [page 55]
PROPERTY Function [System]
DB_PROPERTY Function [System]
CONNECTION_PROPERTY Function [System]
on_charset_conversion_failure Option [page 782]

1.5.4.4 Locales

Both the database server and the client library recognize their language and character set environment using a **locale definition**.

The application locale, or client locale, is used by the client or client library when making requests to the database server to determine the character set in which results should be returned, and the language of error messages, warnings, and other messages. The database server compares its own locale with the application locale to determine whether character set conversion is needed. Different databases on a server may have different locale definitions, and each client may have its own locale.

The locale consists of the following components:

Language

The language is a two-character string using the ISO 639-1 standard values (for example, DE for German). Both the database server and the client have language values for their locale.

The database server uses the locale language to determine the language libraries to load. When creating a database, if no collation is specified, the database server also uses the language, together with the character set, to determine which collation to use.

The client library uses the locale language to determine the language libraries to load, and the language to request from the database.

Character set

The character set is the code page, or encoding, in use. The client and server both have character set values, and they may differ. If they differ, character set conversion is used to enable interoperability.

In this section:

1.5.4.4.1 Locale Language

The locale language is the language being used by the user of the client application, or expected to be used by users of the database server.

The client library, and the database server, both determine the language component of the locale in the same manner:

1. Use the value of the SALANG environment variable, if it exists.
2. For Windows, if the SALANG environment variable doesn't exist, check the SQL Anywhere language registry entry.
3. Check the operating system language setting.
4. If the language still cannot be determined by the above settings, default to English.

Related Information

Locale Information [page 616]
SALANG Environment Variable [page 576]
Registry Settings on Installation [page 592]

1.5.4.4.2 Language Label Values and Language Codes

The following table displays the valid language label values, together with the equivalent ISO 639 language codes:

Language	ISO 639-1 language code	Language label	Alternative label
Arabic	AR	arabic	N/A
Czech	CS	czech	N/A
Danish	DA	danish	N/A
Dutch	NL	dutch	N/A
English	EN	us_english	english
Finnish	FI	finnish	N/A
French	FR	french	N/A
German	DE	german	N/A
Greek	EL	greek	N/A
Hebrew	HE	hebrew	N/A
Hungarian	HU	hungarian	N/A
Italian	IT	italian	N/A
Japanese	JA	japanese	N/A
Korean	KO	korean	N/A
Lithuanian	LT	lithuanian	N/A
Norwegian	NO	norwegian	norweg
Polish	PL	polish	N/A
Portuguese	PT	portuguese	portugue
Russian	RU	russian	N/A
Simplified Chinese	ZH	chinese	simpchin
Spanish	ES	spanish	N/A
Swedish	SV	swedish	N/A
Thai	TH	thai	N/A
Traditional Chinese	TW	tchinese	tradchin
Turkish	TR	turkish	N/A
Ukrainian	UK	ukrainian	N/A

1.5.4.4.3 Locale Character Set

Both application and server locale definitions have a character set.

The application uses its character set when requesting character strings from the database server. The database server compares the database character set with that of the application to determine whether

character set conversion is needed. If the database server cannot convert to and from the client character set, the connection fails.

1. If the SACHARSET environment variable is set, its value is used to determine the character set.
   The database server uses SACHARSET only when creating new databases, and then only if no collation is specified.
2. If the connection string specifies a character set, it is used.
3. SAP Open Client applications check the `locales.dat` file in the `locales` subdirectory of the `SAP` release directory.
4. Character set information from the operating system is used to determine the locale:
   - On Windows operating systems, the current Windows ANSI code page is used.
   - On UNIX and Linux platforms, the following Locale Environment Variables are examined, in the specified order: LC_ALL, LC_MESSAGES, LC_CTYPE, LANG. For the first of these environment variables found to be set, its value is used to determine the character set. If the character set cannot be determined from the operating system, the default of iso_1 (also referred to as Windows code page 28591, ISO 8859-1 Latin I, ISO 8859-1 Latin-1, or iso_8859-1:1987) is used.
5. On any other platform, a default code page cp1252 is used.

Related Information

Locale Information [page 616]
CharSet (CS) Connection Parameter [page 55]
SACHARSET Environment Variable [page 574]

1.5.4.4.4 Locale Information

You can determine locale information using functions such as PROPERTY, DB_PROPERTY, and CONNECTION_PROPERTY.

The following table shows how to use these functions to return locale information about the client connection, database, and database server.

System function and parameter	Return value
`SELECT PROPERTY( 'CharSet' );`	Character set of the database server. Usually the character set of the computer hosting the database server.
`SELECT PROPERTY( 'DefaultCollation' );`	Default CHAR collation used by the database server for creating databases.
`SELECT PROPERTY( 'DefaultNcharCollation' );`	Default NCHAR collation used by the database server for creating databases.
`SELECT PROPERTY( 'Language' );`	The locale language for the database server.

System function and parameter	Return value
`SELECT DB_PROPERTY( 'CharSet' );`	Character set used to store CHAR data in the database.
`SELECT DB_PROPERTY( 'NcharCharSet' );`	Character set used to store NCHAR data in the database.
`SELECT DB_PROPERTY( 'MultiByteCharSet' );`	Whether CHAR data uses a multibyte character set (On=yes, Off=no).
`SELECT DB_PROPERTY( 'Language' );`	Comma-separated list of two-letter codes representing the languages supported by database CHAR collation.
`SELECT DB_PROPERTY( 'Collation' );`	CHAR collation name in use by the database server.
`SELECT DB_PROPERTY( 'NcharCollation' );`	NCHAR collation name in use by the database server.
`SELECT CONNECTION_PROPERTY( 'CharSet' );`	Client's CHAR data character set.
`SELECT CONNECTION_PROPERTY( 'NcharCharSet' );`	Character set of NCHAR data for the connection.
`SELECT CONNECTION_PROPERTY( 'Language' );`	Client language for the connection.

Related Information

PROPERTY Function [System]
DB_PROPERTY Function [System]
CONNECTION_PROPERTY Function [System]

1.5.4.4.5 Setting a Locale

Set a locale that will be used by the database server and client library to recognize the environment.

Context

The locale consists of a language and character set. If the default locale is appropriate for your needs, you do not need to take any action.

Procedure

If you need to change the locale, you can set either or both of the SACHARSET and SALANG environment variables:

```
SACHARSET=charset
SALANG=language-code
```

The `charset` is a valid character set label, and `language-code` is a two-letter language code from the list of valid language codes (for example, EN, DE, or ZH).

Results

The locale is set.

Related Information

Locale Information [page 616]
Supported Character Sets [page 605]
Collation Considerations [page 627]
Language Label Values and Language Codes [page 615]
SACHARSET Environment Variable [page 574]
SALANG Environment Variable [page 576]

1.5.4.5 Collations

A collation describes how to sort and compare characters from a particular character set or encoding.

Two collation algorithms are supported: the SQL Anywhere Collation Algorithm (SACA), and the Unicode Collation Algorithm (UCA). SACA provides fast, compact, and reasonable sorting at the expense of linguistic correctness. UCA provides linguistic correctness, but with a small expense in storage requirements and execution time. UCA also supports collation tailoring options to control how collation is performed.

You set the collation for a database at database creation. You can change the collation later, however.

Collations in a Database

The SQL Anywhere Collation Algorithm (SACA) provides reasonable comparison, ordering, and case conversion of single-byte and multibyte character sets.

The algorithm is space efficient and fast. The mapped form of a string, such as an index, is the same length as the original string. The mappings for comparison, ordering, and case conversion use a simple table lookup of each byte value of the string.

Single-Byte Character Sets

In a typical collation for a single-byte character set, all accented and unaccented forms of a character are mapped to the same value, making the collation accent insensitive. Accented and unaccented forms of the same letter compare as exactly equal and sort near each other.

The collation also provides conversion between uppercase and lowercase letters, preserving accents.

Multibyte Character Sets

In multibyte character sets, the lead-bytes are mapped into the 256 distinct values. Follow bytes are compared using their binary value.

For most collations for multibyte character sets, this mapping technique provides a reasonable ordering because the character set encoding groups characters into 256-byte pages identified by the lead byte. The pages, and the characters within each page, are in a reasonable order in the character set. The collations typically preserve the ordering of the pages (lead bytes) within the character set. Some pages may be ordered by other characteristics. For example, the 932JPN collation provided for Japanese code page 932 groups the full-width (Kanji) and half-width (katakana) characters.

Case conversion is provided only for the 7-bit English characters.

UTF-8 Character Sets

UTF-8 is a multibyte character set. Each character contains from one to four bytes. The UTF8BIN collation is provided for sorting UTF-8 characters.

In UTF8BIN, lead bytes are mapped into 256 distinct values, and follow bytes are compared using their binary values. Because of the representation of characters in UTF-8 and the limitation of 256 distinct mapping values, it is not possible to group related characters such as accented and unaccented forms of the same letter. The ordering is essentially binary.

Case conversion is supported only for the 7-bit English characters.

In this section:

Collations in a Database [page 620]
There are several collations in a database.

SQL Anywhere Collation Algorithm (SACA) [page 621]
The SQL Anywhere Collation Algorithm (SACA) provides reasonable comparison, ordering, and case conversion of single-byte and multibyte character sets.

Unicode Collation Algorithm (UCA) [page 622]
The Unicode Collation Algorithm (UCA) is an algorithm for sorting the entire Unicode character set.

Alternate Collations [page 624]
There are several compatibility collations that can be used with the SORTKEY and COMPARE functions.

Collation Considerations [page 627]
You specify the collation for a database when you create the database.

How the Database Server Chooses the Default Collation for a New Database [page 631]
When a new database is created, and the collation is not explicitly specified, the database server uses the language and character set to determine the collation.

Changing the Database Collation [page 632]
Change your database collation from one collation to another by performing an unload and reload into the collation of choice.

Related Information

Database Creation [page 277]
Initialization Utility (dbinit) [page 1166]
Collation Tailoring Options [page 628]

1.5.4.5.1 Collations in a Database

There are several collations in a database.

CHAR Collation

CHAR data types, including CHAR, VARCHAR, and LONG VARCHAR, can use a collation that uses the SQL Anywhere Collation Algorithm (SACA) or they can use the Unicode Collation Algorithm (UCA). In either case, the collation used is referred to as the CHAR collation.

NCHAR Collation

NCHAR data types, including NCHAR, NVARCHAR, and LONG NVARCHAR, can use the Unicode Collation Algorithm (UCA) or can use the UTF8BIN collation, which uses the SQL Anywhere Collation Algorithm (SACA).

Choosing Case and Accent Sensitivity

When a database is created, if case sensitivity is not specified, then it is case insensitive. The database can be made case sensitive by specifying the appropriate option. It is not possible to change the case sensitivity after the database has been created without rebuilding the database.

The case sensitivity for the database determines the case sensitivity for both the SACA and UCA collations, and so it also determines the case sensitivity of both the CHAR and NCHAR collations.

When a database is created, if accent sensitivity is not specified, then it is accent insensitive. The database can be made accent sensitive by specifying the appropriate option. It is not possible to change the accent sensitivity after the database has been created without rebuilding the database.

The accent sensitivity for the database affects only the UCA collation, whether it is used for the CHAR or NCHAR collations or both. If you choose SACA collations for both CHAR and NCHAR collations, then the options for accent sensitivity have no effect. Accent sensitivity is an attribute of SACA collations and cannot be specified using the options provided when creating the database.

1.5.4.5.2 SQL Anywhere Collation Algorithm (SACA)

The SQL Anywhere Collation Algorithm (SACA) provides reasonable comparison, ordering, and case conversion of single-byte and multibyte character sets.

The algorithm is space efficient and fast. The mapped form of a string, such as an index, is the same length as the original string. The mappings for comparison, ordering, and case conversion use a simple table lookup of each byte value of the string.

Single-Byte Character Sets

In a typical collation for a single-byte character set, all accented and unaccented forms of a character are mapped to the same value, making the collation accent insensitive. Accented and unaccented forms of the same letter compare as exactly equal and sort near each other.

The collation also provides conversion between uppercase and lowercase letters, preserving accents.

Multibyte Character Sets

In multibyte character sets, the lead-bytes are mapped into the 256 distinct values. Follow bytes are compared using their binary value.

For most collations for multibyte character sets, this mapping technique provides a reasonable ordering because the character set encoding groups characters into 256-byte pages identified by the lead byte. The pages, and the characters within each page, are in a reasonable order in the character set. The collations typically preserve the ordering of the pages (lead bytes) within the character set. Some pages may be ordered

by other characteristics. For example, the 932JPN collation provided for Japanese code page 932 groups the full-width (Kanji) and half-width (katakana) characters.

Case conversion is provided only for the 7-bit English characters.

UTF-8 Character Sets

UTF-8 is a multibyte character set. Each character contains from one to four bytes. The UTF8BIN collation is provided for sorting UTF-8 characters.

In UTF8BIN, lead bytes are mapped into 256 distinct values, and follow bytes are compared using their binary values. Because of the representation of characters in UTF-8 and the limitation of 256 distinct mapping values, it is not possible to group related characters such as accented and unaccented forms of the same letter. The ordering is essentially binary.

Case conversion is supported only for the 7-bit English characters.

1.5.4.5.3 Unicode Collation Algorithm (UCA)

The Unicode Collation Algorithm (UCA) is an algorithm for sorting the entire Unicode character set.

It provides linguistically correct comparison, ordering, and case conversion. The UCA was developed as part of the Unicode standard. The UCA is implemented using the International Components for Unicode (ICU) open source library, developed and maintained by IBM.

> **i** Note
>
> The default UCA ordering sorts most characters in most languages into an appropriate order. However, because of the sorting and comparison variations between languages sharing characters, the UCA cannot provide proper sorting for all languages. For this purpose, ICU provides collation tailoring options to the UCA.

The UCA provides advanced comparison, ordering, and case conversion at a small cost in space and time.

The mapped form of a string is longer than the original string. The algorithm provides sophisticated handling of more complex characters.

Unlike the SQL Anywhere Collation Algorithm (SACA) the Unicode Collation Algorithm (UCA) is only for use with single-byte and UTF-8 character sets, and it separates each character into one or more attributes. For letters, these attributes are base character, accent, and case.

Non-letters typically have only one attribute, the base character.

UCA compares character strings as follows:

- Compare the base characters. If one string of base characters differs from the other, then the comparison is complete. Accent and case are not considered.
- If the database is accent sensitive, compare the accents. If the accents differ, then the comparison is complete. Case is not considered.

- If the database is case sensitive, compare the case of each character.

The original string values are equal if and only if the base characters, accents, and case are the same for both strings.

Example

Suppose UCA is used to compare the strings in the first column of the table below. The subsequent columns describe the three attributes for each string. The base characters are identical; the words differ only in accents and case.

String	Base characters	Accents	Case
noel	noel	none, none, none, none	lower, lower, lower, lower
noël	noel	none, none, accent, none	lower, lower, lower, lower
Noel	noel	none, none, none, none	upper, lower, lower, lower
Noël	noel	none, none, accent, none	upper, lower, lower, lower

The following table shows the ordering that would occur in the four possible combinations of accent- and case-sensitivity using UCA:

Accent sensitive	Case sensitive	ORDER BY result	Explanation
N	N	Noel, noël, Noël, noel in any order	• Accents ignored • Case ignored • All values considered equal • Random order within set of four
Y	N	Noel, noel in any order, followed by noël, Noël in any order	• No-accents before accents, so e before ë • Case ignored, N and n are in random order within each set of two
N	Y	Noel, Noël in any order, followed by noël, noel in any order	• Uppercase before lowercase, so N before n • Accents ignored, e and ë are in random order within each set of two
Y	Y	Noel noel Noël noël	• No-accents before accents, so e before ë • Uppercase before lowercase, so N before n

Related Information

Collation Tailoring Options [page 628]

1.5.4.5.4 Alternate Collations

There are several compatibility collations that can be used with the SORTKEY and COMPARE functions.

Collation label	Description
874THAIBIN	Code Page 874, Windows Thai, ISO8859-11, binary ordering
932JPN	Code Page 932, Japanese Shift-JIS with Microsoft extensions
936ZHO	Code Page 936, Simplified Chinese, PRC GBK 2312-80 8-bit encoding
949KOR	Code Page 949, Korean KS C 5601-1987 encoding, Wansung
950ZHO_HK	Code Page 950, Traditional Chinese, Big 5 encoding with HKSCS
950ZHO_TW	Code Page 950, Traditional Chinese, Big 5 encoding
1250LATIN2	Code Page 1250, Windows Latin 2, Central/Eastern European
1250POL	Code Page 1250, Windows Latin 2, Polish
1251CYR	Code Page 1251, Cyrillic
1252BIN	Code Page 1252, Windows Latin 1, binary ordering
1252LATIN1	Code Page 1252, Windows Latin 1, Western
1252LT1ACC	Code Page 1252, Windows Specialty Latin 1, Western, accented chars not equal
1252NOR	Code Page 1252, Windows Latin 1, Norwegian
1252SPA	Code Page 1252, Windows Latin 1, Spanish
1252SWEFIN	Code Page 1252, Windows Latin 1, Swedish/Finnish
1253ELL	Code Page 1253, Windows Greek, ISO8859-7 with extensions
1254TRK	Code Page 1254, Windows Latin 5, Turkish, ISO 8859-9 with extensions
1254TRKALT	Code Page 1254, Windows Turkish, ISO8859-9 with extensions, I-dot equals I-no-dot
1255HEB	Code Page 1255, Windows Hebrew, ISO8859-8 with extensions
1256ARA	Code Page 1256, Windows Arabic, ISO8859-6 with extensions
1257LIT	Code Page 1257, Lithuanian

Collation label	Description
CESU8BIN	CESU-8, 8-bit multibyte encoding for Unicode, binary ordering
EUC_CHINA	Simplified Chinese GB 2312-80 Encoding
EUC_JAPAN	Japanese EUC JIS X 0208-1990 and JIS X 0212-1990 Encoding
EUC_KOREA	Korean KS C 5601-1992 Encoding, Johab
EUC_TAIWAN	Taiwanese Big 5 Encoding
ISO1LATIN1	ISO8859-1, ISO Latin 1, Western, Latin 1 Ordering
ISO9LATIN1	ISO8859-15, ISO Latin 9, Western, Latin 1 Ordering
ISO_1	ISO8859-1, Latin 1, Western
ISO_BINENG	Binary ordering, English ISO/ASCII 7-bit letter case mappings
UCA	Standard default UCA collation
UTF8BIN	UTF-8, 8-bit multibyte encoding for Unicode, binary ordering

Alternate Collations

Alternate collations are available for compatibility with older versions of SQL Anywhere, or for special purposes. To see the full list of supported alternate collations, run the following command:

```
dbinit -l+
```

SAP Adaptive Server Enterprise collations

The following table lists the supported SAP Adaptive Server Enterprise collations for use with features such as the SORTKEY function.

Description	Collation name	Collation ID
Default Unicode multilingual	default	0
CP 850 Alternative: no accent	altnoacc	39
CP 850 Alternative: lowercase first	altdict	45
CP 850 Western European: no case, preference	altnocsp	46
CP 850 Scandinavian dictionary	scandict	47
CP 850 Scandinavian: no case, preference	scannocp	48

Description	Collation name	Collation ID
GB Pinyin	gbpinyin	n/a
Binary sort	binary	50
Latin-1 English, French, German dictionary	dict	51
Latin-1 English, French, German no case	nocase	52
Latin-1 English, French, German no case, preference	nocasep	53
Latin-1 English, French, German no accent	noaccent	54
Latin-1 Spanish dictionary	espdict	55
Latin-1 Spanish no case	espnocs	56
Latin-1 Spanish no accent	espnoac	57
ISO 8859-5 Russian dictionary	rusdict	58
ISO 8859-5 Russian no case	rusnocs	59
ISO 8859-5 Cyrillic dictionary	cyrdict	63
ISO 8859-5 Cyrillic no case	cyrnocs	64
ISO 8859-7 Greek dictionary	elldict	65
ISO 8859-2 Hungarian dictionary	hundict	69
ISO 8859-2 Hungarian no accents	hunnoac	70
ISO 8859-2 Hungarian no case	hunnocs	71
ISO 8859-5 Turkish dictionary	turdict	72
ISO 8859-5 Turkish no accents	turnoac	73
ISO 8859-5 Turkish no case	turnocs	74
CP 874 (TIS 620) Royal Thai dictionary	thaidict	1
ISO 14651 ordering standard	14651	22
Shift-JIS binary order	sjisbin	179
Unicode UTF-8 binary sort	utf8bin	24
EUC JIS binary order	eucjisbn	192
GB2312 binary order	gb2312bn	137
CP932 MS binary order	cp932bin	129
Big5 binary order	big5bin	194
EUC KSC binary order	euckscbn	161

Related Information

What Is ICU and When Is It Needed? [page 599]

COMPARE Function [String]
SORTKEY Function [String]

1.5.4.5.5 Collation Considerations

You specify the collation for a database when you create the database.

The default collation is inferred from the code page and language of the database server's computer's operating system. Usually the default collation is a suitable choice, but you can also explicitly specify a collation to match your needs from the wide selection of supplied collations. More than one collation for a particular language can be supported.

You should choose a collation that uses a character set and sort order that are appropriate for the data in your database. You can also specify collation tailoring options for additional control over the sorting and comparing of characters.

When choosing the collation for your database, consider the following:

- There is a performance cost, and extra complexity in system configuration, when you use character set conversion. Choose a collation that avoids the need for character set conversion. Character set conversion is not used if the database server and client use the same character set.
  You can avoid character set conversion by using a collation sequence in the database that matches the character set in use on your client computer operating system. Choose the ANSI character set for Windows operating systems on the client computer.
- If your client computers use a variety of characters sets, or if the database must store Unicode data, consider using the UCA and/or UTF8BIN collations. However, the UCA collation cannot be used with multibyte character sets other than UTF-8.
- Choose a collation that uses a character set and sort order appropriate for the data in the database. It is often the case that there are several collations that meet this requirement.

In this section:

Related Information

Database Creation [page 277]
Creating a Database (SQL Central) [page 278]
Creating a Database (dbinit Utility) [page 279]
Recommended Character Sets and Collations [page 634]
CREATE DATABASE Statement
Initialization Utility (dbinit) [page 1166]

1.5.4.5.5.1 Collation Tailoring Options

If you choose the UCA collation when you create a database, you can optionally specify collation tailoring options.

If you do not choose UCA as the collation, you can still use tailoring syntax to control case sensitivity. You can also specify tailoring options when comparing or sorting data using the COMPARE and SORTKEY functions.

Swedish Collation Tailoring

To tailor a UCA collation to conform to the Swedish Academy's 2005 standards in which V and W are considered to be different characters at the primary level, specify `UCA (locale=swe;sorttype=phonebook)`. Without sorttype=phonebook, V and W are considered to be the same character in the Swedish locale.

Japanese Collation Tailoring

UCA distinguishes some Hiragana and Katakana letters only at the tertiary level and those differences are lost in case insensitive collations. To tailor a UCA collation to define primary level differences between all Hiragana letters, as well as primary-level differences between all Katakana letters, specify `UCA (locale=ja;sorttype=direct;...)`. Although these tailoring options do not provide absolute correct sorting semantics, they do provide correct equality semantics.

Summary of Collation Tailoring Options

Collation tailoring options take the form of keyword-value pairs. Following is a table of the supported keywords, including their allowed alternate forms, and their allowed values.

> **i** Note
>
> Databases created with collation tailoring options cannot be started using a version 10.0.0 or earlier database server.

Keyword	Collation	Alternate forms	Allowed values
Locale	UCA	(none)	Any valid locale code. For example, en.

Keyword	Collation	Alternate forms	Allowed values
CaseSensitivity	All supported collations	CaseSensitive, Case	**respect** Respect case differences between letters. For the UCA collation, this is equivalent to UpperFirst. For other collations, it depends on the collation itself. **ignore** Ignore case differences between letters. **UpperFirst** Always sort uppercase first (Aa). **LowerFirst** Always sort lowercase first (aA).
AccentSensitivity	UCA	AccentSensitive, Accent	**respect** Respect accent differences between letters. **ignore** Ignore accent differences between letters. **French** Respect accent sensitivity with French rules.

Keyword	Collation	Alternate forms	Allowed values
PunctuationSensitivity	UCA	PunctuationSensitive, Punct	**ignore** Ignore differences in punctuation. **primary** Use first level sorting (consider letter, only). For example, a > b. **quaternary** Use fourth level sorting: consider letter first, then case, then accent, and then punctuation. For example, multiByte, multibyte, multi-byte, and multi-Byte, are sorted as: • multiByte • multibyte • multi-Byte • multi-byte You cannot specify quaternary with a case- or accent-insensitive database.
SortType	UCA	(none)	The type of sort to use. Possible values include: • phonebook • traditional • standard • pinyin • stroke • direct • posix • big5han • gb2312han For more information about these sort types, see Unicode Technical Standard #35 .

Related Information

CREATE DATABASE Statement
Initialization Utility (dbinit) [page 1166]
COMPARE Function [String]
SORTKEY Function [String]

1.5.4.5.6 How the Database Server Chooses the Default Collation for a New Database

When a new database is created, and the collation is not explicitly specified, the database server uses the language and character set to determine the collation.

- The language comes from the SALANG environment variable (if it exists), the registry, or the operating system.
- The character set comes from the SACHARSET environment variable (if it exists) or the operating system.

In this section:

Related Information

SALANG Environment Variable [page 576]
SACHARSET Environment Variable [page 574]

1.5.4.5.6.1 Determining the Default Collation

Determine the default collation.

Context

A collation describes how to sort and compare characters from a particular character set or encoding. If you do not explicitly specify a collation when creating a database a default collation is used, depending on the operating system.

Procedure

1. Connect to the database.
2. Execute the following statement:

```
SELECT PROPERTY( 'DefaultCollation' );
```

Results

The default collation for the database is returned.

Related Information

Collation Considerations [page 627]
Character Sets [page 601]

1.5.4.5.7 Changing the Database Collation

Change your database collation from one collation to another by performing an unload and reload into the collation of choice.

Prerequisites

You must have the SELECT ANY TABLE and SERVER OPERATOR system privileges.

By default, you must have the SELECT ANY TABLE system privilege to execute an UNLOAD statement. The required privileges can be changed by using the -gl database server option.

Context

Collations are chosen at database creation time and cannot be changed without rebuilding the database.

Procedure

1. Start the database:

   ```
   START databasefile
   ```

2. Determine the character set for the existing database:

   ```
   SELECT DB_PROPERTY( 'CharSet' );
   ```

   For early versions of SQL Anywhere, this property may not exist. The character set is also implied by the collation name. For example, collation 1252LATIN1 uses code page 1252.

   The character set in the existing database should be the same as, or compatible with, the operating system and client character sets. If it is not, it is an excellent reason to rebuild the database, but requires great care in the rebuild process.

   In particular, if you have been using a database with collation 850LATIN1 with earlier versions of SQL Anywhere that either did not support character set conversion (versions 5 and earlier) or disabled it by default (versions 6 and 7), and if your client applications were normal Windows applications, you may have code page 1252 character data in your database that is interpreting data to be in code page 850.

3. Determine the character set for the data in the existing database:

   ```
   UNLOAD TABLE mytable TO 'mytable-data-in-utf8.dat' ENCODING 'UTF-8';
   ```

   View the results in a text editor.

   If accented data is correct, then the character data in the database matches the Windows ANSI code page, which for English and other Western European languages is code page 1252. If the data appears correct in a DOS-based editor, then the character data matches the Windows OEM code page, which is likely 437 or 850.

4. Unload the database.

   If the data character set is incompatible with the database character set, it is critical that the data is unloaded without character set conversion. Depending on the version of SQL Anywhere being used, you can use the internal unload feature of dbunload, or manually unload the data using the UNLOAD TABLE statement.

5. Create the new database, specifying the collations and character sets you want to use, and set the DBA user ID and password to DBA and passwd:

   ```
   dbinit -dba DBA,passwd -z 1252LATIN1 c:\newdatabase.db
   ```

6. Stop the old database server and start the new database server:

   ```
   dbsrv17 -n new-server c:\newdatabase.db
   ```

7. Load the data into the new database.

   If the unloaded data and schema (`reload.sql`) match the character set of the computer used to perform the reload, you can use the external reload option of dbunload. The data is automatically converted to the correct character set for the database. For example:

   ```
   dbunload -ix c:\databasefile
   ```

If the data's encoding does not match the character set of the database, and you are loading the data using LOAD TABLE statements (internal reload), you must use the ENCODING clause; the database server does not, by default, perform character set conversion for data loaded using LOAD TABLE statements.

If the data's encoding does not match the code page of the computer on which you are working, and you are loading using INPUT statements (external reload), you must use the ENCODING clause; otherwise, the database server assumes that the data is in the computer's native character set.

Results

The database is recreated with the new collation.

Example

For example, suppose you have a database with collation ISO1LATIN1 but discovered you need to support the euro currency symbol. Since ISO1LATIN1 does not support the euro symbol, but other European character sets such as 1252LATIN1 do, you can unload your database, create a new one with 1252LATIN1, and reload your database.

Related Information

Collation Considerations [page 627]
Database Rebuilds
LOAD TABLE Statement
UNLOAD Statement
INPUT Statement [Interactive SQL]
Unload Utility (dbunload) [page 1233]
CharSet (CS) Connection Parameter [page 55]
-gl Database Server Option [page 434]
Changing the Collation and Codepage When Rebuilding an SQL Anywhere Database

1.5.4.6 Recommended Character Sets and Collations

While the names of hundreds of character sets, code pages, encodings, and collations are recognized, there are several that are recommended for use with Windows and UNIX/Linux platforms, depending on the language in use.

Use the dbinit -le option to obtain a list of all the available character set labels for a database. Use the dbinit -l option to obtain a list of available collations for a database.

Where a character set encoding or label must be specified, use the value from the Character set label column or one of the labels listed by dbinit -le. Where a collation must be specified, use the value from the Collation or Alternate collation column or one of the labels listed by dbinit -l.

> **i** Note
>
> For languages not found in the tables below, the UTF-8 encoding should be used with collations UCA or UTF8BIN.

Microsoft Windows Platforms

Language	Windows Code Page	Character set label	Collation	Alternate collation
Arabic	1256	Windows-1256	1256ARA	
Central and Eastern European	1250	Windows-1250	1250LATIN2	
Danish	1252	Windows-1252	1252LATIN1	
Dutch	1252	Windows-1252	1252LATIN1	
English	1252	Windows-1252	1252LATIN1	
Finnish	1252	Windows-1252	1252SWEFIN	
French	1252	Windows-1252	1252LATIN1	
German	1252	Windows-1252	1252LATIN1	
Greek	1253	Windows-1253	1253ELL	
Hebrew	1253	Windows-1253	1255HEB	
Italian	1252	Windows-1252	1252LATIN1	
Japanese	932	Windows-31J	932JPN	
Korean	949	IBM949	949KOR	
Lithuanian	1257	Windows-1257	1257LIT	
Norwegian	1252	Windows-1252	1252NOR	
Polish	1250	Windows-1250	1250POL	
Portuguese	1252	Windows-1252	1252LATIN1	
Russian	1251	Windows-1251	1251CYR	
Simplified Chinese	936	GBK	936ZHO	
Spanish	1252	Windows-1252	1252SPA	
Swedish	1252	Windows-1252	1252SWEFIN	
Thai	874	TIS-620	874THAIBIN	
Traditional Chinese - Hong Kong	950	Big5-HKSCS	950ZHO_HK	

Language	Windows Code Page	Character set label	Collation	Alternate collation
Traditional Chinese - Taiwan	950	Big5	950ZHO_TW	
Turkish	1254	Windows-1254	1254TRK	1254TRKALT
Ukrainian	1251	Windows-1251	1251CYR	
Western European	1252	Windows-1252	1252LATIN1	

UNIX/Linux Platforms

Language	Character set label	Collation	Alternate collation
Arabic	ISO_8859-6:1987	UCA	
Central and Eastern European	ISO_8859-2:1987	UCA	
Danish	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
Dutch	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
English	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
Finnish	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
French	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
German	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
Greek	ISO_8859-7:1987	UCA	
Hebrew	ISO_8859-8:1988	UCA	
Italian	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
Japanese	EUC-JP[1]	EUC_JAPAN	
Korean	EUC-KR	EUC_KOREA	
Lithuanian	(use UTF-8)	UCA or UTF8BIN	
Norwegian	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
Polish	ISO_8859-2:1987	UCA	
Portuguese	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
Russian	ISO_8859-5:1988	UCA	
Simplified Chinese	GB2312	EUC_CHINA	
Spanish	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
Swedish	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
Thai	(use UTF-8)	UCA or UTF8BIN	
Traditional Chinese - Hong Kong	Big5-HKSCS	950ZHO_HK	950TWN
Traditional Chinese - Taiwan	EUC-TW	EUC_TAIWAN	

Language	Character set label	Collation	Alternate collation
Traditional Chinese - Taiwan	Big5	950ZHO_TW	
Turkish	ISO_8859-9:1989	920TRK	
Ukrainian	ISO_8859-5:1988	UCA	
Western European	ISO-8859-15	ISO9LATIN1	ISO1LATIN1

[1] EUC-JP is an alternate label for the SQL Anywhere label Extended_UNIX_Code_Packed_Format_for_Japanese.

Related Information

Initialization Utility (dbinit) [page 1166]

1.5.4.7 Turkish Character Sets and Collations

The Turkish language has two forms of what appears to be the letter *I*.

One form, referred to as I-dot, appears as the following:

```
i, İ
```

The second form, referred to as I-no-dot, appears as the following:

```
ı, I
```

Even though these letters appear as variations of the same letter, in the Turkish alphabet they are considered to be distinct letters. The Turkish collation 1254TRK is provided to support these variations.

Turkish rules for case conversion of these characters are incompatible with ANSI SQL standard rules for case conversion. For example, Turkish says that the lowercase equivalent of `I` is:

```
ı
```

However, the ANSI standard says that it is:

```
i
```

For this reason, correct case-insensitive matching is dependent on whether the text being matched is Turkish or English/ANSI. In many contexts, there is not enough information to make such a distinction, which leads to some non-standard behaviors in such databases.

For example, consider the following statements, executed against a database using the 1254TRK collation:

```
SELECT * FROM syshistory   //actual table name is SYSHISTORY
SELECT * FROM fig          //actual name is FİG
```

The first statement references a system object, and ANSI SQL conversion rules are required to match the name. The second statement references a user object, and Turkish conversion rules are required to match the name. However, the database server cannot tell which conversion rules to use until it knows what the object is, and it cannot know what the object is until it knows what conversion rules to use. The situation cannot be resolved properly for both system and user objects. In this example, since the database server is using the Turkish collation 1254TRK, the first statement fails because lowercase I is not considered equivalent to uppercase I. The second statement succeeds.

The incompatibility of Turkish and ANSI standards requires that system object references in Turkish databases specify the object name in the correct case, that is, the case used to create the object. The first statement above should be written as follows:

```
SELECT * FROM SYSHISTORY
```

In fact, only the letter I must be in the correct case.

As an alternative, it is acceptable, although unusual, to write the statement as follows:

```
SELECT * FROM syshıstory   //I-no-dot
```

Keywords such as INSERT are case-insensitive, even in Turkish databases. SQL Anywhere knows that all keywords use only English letters, so it uses ANSI case conversion rules when matching keywords. This knowledge is also applied for certain other identifiers, such as built-in functions. However, objects whose names are stored in the catalog must be specified using the correct case or letter, as described above. To determine the correct case, look in the system view that defines the system object.

In this section:

Related Information

System Views

1.5.4.7.1 Data in Case-insensitive Turkish Databases

There are several rules governing data in case-insensitive Turkish databases.

For example if a data value is:

```
FİG
```

then a lowercase reference to that data should be:

```
fig
```

Then, the same I-dot character is used in both forms.

1.5.4.7.2 Alternative Turkish Collation 1254TRKALT

For some application developers, the Turkish letter I problem can cause significant problems.

The correct solution is to ensure that all object references are in the proper case or that the proper form of the letter I is used. However, it may be more expedient to violate the Turkish rules in favor of the ANSI rules.

The collation 1254TRKALT is provided. This collation is identical to 1254TRK, except that it makes I-dot and I-no-dot equivalent characters.

It is important to understand the consequences of this change. In a 1254TRKALT database, the following strings are equal:

```
fig
fıg
```

This is not correct for a Turkish user, but may be acceptable for other users.

The second issue appears when using ORDER BY. Consider the following strings:

```
ia
ıa
ıs
is
```

In a 1254TRK database, an ORDER BY of the strings would produce the following:

```
ıa
ıs
ia
is
```

This is because I-no-dot is less than I-dot. In a 1254TRKALT database, the order would be:

```
ia
ıa
ıs
is
```

This is because I-no-dot is equal to I-dot.

1.5.5 Login Policies

A **login policy** consists of a set of rules that are applied when you create a database connection for a user.

Login policies govern only the rules for user login and are separate from roles and privileges. Login policies are not inherited. All users in a database are assigned a login policy.

You can create, edit, and delete login policies. As well, you can assign login policies to users.

Use sa_get_user_status system procedure to view information about the current status, including the login policy, of a user.

In this section:

Related Information

CREATE LOGIN POLICY Statement
ALTER LOGIN POLICY Statement
DROP LOGIN POLICY Statement
sa_get_user_status System Procedure

1.5.5.1 Root Login Policy and Inheritance

All new databases include the *root* login policy. You can modify the values for the root login policy options, but you cannot delete the policy.

A user is assigned the root login policy when:

- you create a new user and do not specify a login policy
- you use the Unload utility (dbunload) to rebuild a database created by a previous version of SQL Anywhere
- you upgrade a SQL Anywhere version 10 database using the Upgrade utility (dbupgrad) or the ALTER DATABASE UPGRADE statement

Inheritance of Login Policy Settings

A default login policy called **root** is stored in the database and contains the default option values for all policies. To use different settings than the defaults, you can either alter the root policy, or create a policy and then alter it to contain overrides for the defaults. A policy inherits its default settings from the root policy, unless it is altered to contain overrides.

For example, suppose the root policy value for max_connections is 5. You create a policy called myPolicy and alter it to set max_connections to Unlimited. Then, you create a user and assign them the myPolicy login policy. When the user logs in, their login policy option settings are inherited from the root login policy with the exception of max_connections, which is set to Unlimited.

Inheritance of default values from the root policy is important to understand because if you subsequently change the value of an option setting in the root policy, you impact users of policies that rely on the default value for that setting. Similarly, if a root value is changed, it does not impact any users of policies that contain an override for that setting.

Related Information

Setting a Dual Control Password (SQL) [page 1639]
ALTER LOGIN POLICY Statement

1.5.5.2 Login Policy Options and Default Values

The following table lists the options that are governed by a login policy and includes the default values for the *root* login policy:

Policy-option-name	Description	Default value	Applies to:
auto_unlock_time	The time period after which locked accounts for users without the MANAGE ANY USER system privilege are automatically unlocked.	Unlimited	All users except those with the MANAGE ANY USER system privilege
change_password_dual_control	When the value for this option is ON, setting the password requires two administrators. The setting for the verify_password_function option is ignored if this option is set to ON because the password is configured separately in two parts. No verification is performed.	OFF	All users
ldap_primary_server	The name of the primary LDAP server.	(none)	All users
ldap_secondary_server	The name of the secondary LDAP server.	(none)	All users
ldap_auto_failback_period	The time period in minutes after which automatic failback to primary server is attempted.	15 minutes	All users
ldap_failover_to_std	This option controls whether the database server permits Standard authentication when authentication with the LDAP server fails due to failure to locate the Distinguished Name (DN) for a user, lack of system resources, network outage, connection timeouts, or similar system failures. This setting does not permit an actual authentication failure returned from an LDAP server to fail over to Standard authentication (as is the case when the user is located but the supplied password does not match).	ON	All users

Policy-option-name	Description	Default value	Applies to:
ldap_refresh_dn	At the time this policy option is specified by a CREATE LOGIN POLICY or ALTER LOGIN POLICY statement, the current time value is stored with the login policy. This value is the timestamp against which user authentication compares the user_dn_cached_at value found for the user in ISYSUSER. If the value in the policy is newer than the user_dn_cached_at value in ISYSUSER, then a search for a user's Distinguished Name (DN) is done to refresh the user_dn value in ISYSUSER. The value NOW is the only valid value to assign to this policy option. All others result in an error. The value is in Coordinated Universal Time (UTC) and is stored as a string in the database server default format.	(none)	All users
locked	If the value for this option is ON, then users are not allowed to establish new connections. The reason_locked column of the sa_get_user_status system procedure returns a string generated by the database server that shows why a user is locked.	OFF	All users except those with the MANAGE ANY USER system privilege
max_connections	The maximum number of concurrent connections allowed for a user with this login policy.	UNLIMITED	All users except those with the SERVER OPERATOR or DROP CONNECTION system privilege

Policy-option-name	Description	Default value	Applies to:
max_failed_login_attempts	The maximum number of failed attempts since the last successful attempt to log in. The user with this login policy is automatically prevented from establishing new connections (locked) when this threshold is exceeded. Users with the SYS_AUTH_DBA_ROLE compatibility role are unlocked after one minute has passed since the most recent failed login attempt.	UNLIMITED	All users
max_days_since_login	The maximum number of days that can elapse between two successive logins by the same user. The user with this login policy is automatically prevented from establishing new connections (locked) when this threshold is exceeded.	UNLIMITED	All users except those with the MANAGE ANY USER system privilege
max_non_dba_connections	The maximum number of concurrent connections that users can make. Users with this login policy are automatically prevented from establishing new connections (locked) when this threshold is exceeded. This option is only supported in the root login policy.	UNLIMITED	All users except those with the SERVER OPERATOR or DROP CONNECTION system privilege

Policy-option-name	Description	Default value	Applies to:
pam_failover_to_std	If the value for this option is ON, then the database server fails over to standard authentication when PAM user authentication cannot perform the authentication. Failover to standard authentication occurs only if this option is set to ON and the following conditions are met: • The database server cannot load the PAM support library (`libpam.so` on most UNIX and Linux systems, `libpam.a` on AIX, and `libpam.dylib` on macOS). • PAM experiences a system or configuration error and cannot authenticate the user.	ON	All users
pam_service_name	The name of the PAM service used to authenticate the database user. This value must be a non-empty, valid alphanumeric ASCII string and refer to the PAM service defined in the computer's PAM configuration file (`/etc/pam.con`) or directory (`/etc/pam.d`).	(none)	All users
password_life_time	The maximum number of days before a password must be changed.	UNLIMITED	All users
password_grace_time	The number of days before the password expires during which login is allowed, but the default post_login procedure issues warnings.	0	All users

Policy-option-name	Description	Default value	Applies to:
password_expiry_on_next_login	If the value for this option is set to ON in the user's assigned login policy and if you use the CREATE USER statement or one of the ALTER USER statement LOGIN POLICY or RESET LOGIN POLICY clauses, then the user's password expires immediately when they next login even if they are assigned to the same policy. If the value for this option is set to OFF in the user's assigned login policy, then you can use the FORCE PASSWORD CHANGE clause of the ALTER USER statement to force the user to change their password when they next login.	OFF	All users
root_auto_unlock_time	The time period after which locked accounts for users with the MANAGE ANY USER system privilege are automatically unlocked. This option is only supported in the root login policy.	1 minute	Users with the MANAGE ANY USER system privilege

The word *UNLIMITED* is a keyword. If a login policy option value is set to something other than the default *UNLIMITED*, then reset it to this default using this keyword. The following is an example.

```
ALTER LOGIN POLICY new_policy
    max_failed_login_attempts=UNLIMITED
    password_expiry_on_next_login=OFF
    locked=OFF;
```

1.5.5.3 Creating a Login Policy

Create custom login policies based on the root login policy.

Prerequisites

You must have the MANAGE ANY LOGIN POLICY system privilege.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, right-click *Login Policies* and then click *New* > *Login Policy*.
3. Follow the instructions in the *Create Login Policy Wizard*.

Results

A new login policy is created.

Next Steps

Assign the new login policy to a user.

Related Information

Assigning a Login Policy to an Existing User [page 647]
Altering a Login Policy [page 648]
CREATE LOGIN POLICY Statement

1.5.5.4 Assigning a Login Policy to an Existing User

Change the login policy that is assigned to a user by assigning the user a new login policy.

Prerequisites

You must have the MANAGE ANY USER system privilege.

Context

All users have a login policy assigned to them. You cannot delete a login policy from a user; you can only change the login policy that is assigned to the user.

Procedure

1. In SQL Central connect to the database.
2. In the left pane, click *Users*.
3. In the right pane, right-click a user and click *Set Login Policy* .
4. In the *Login Policy* list, choose a login policy.

Results

The user's login policy is reassigned. The settings take effect immediately.

Related Information

ALTER USER Statement

1.5.5.5 Altering a Login Policy

Edit a login policy and change its options.

Prerequisites

You must have the MANAGE ANY LOGIN POLICY system privilege.

Context

You can edit any login policy, including the root login policy.

> **i** Note
>
> If you change the value of an option setting in the root login policy, you impact all users that rely on that default value that are assigned the root login policy. Conversely, all users assigned to custom login policies with overrides for that setting are not affected.

Procedure

1. In SQL Central connect to the database.
2. In the left pane, click *Login Policies*.
3. In the right pane, right-click a login policy and click *Properties*.

   To modify the root login policy, right-click *root*, and then click *Properties*.
4. Alter the policy values and then click *OK*.

Results

The login policy is altered. The settings take effect immediately.

Related Information

Assigning a Login Policy to an Existing User [page 647]
Login Policy Options and Default Values [page 642]
ALTER LOGIN POLICY Statement

1.5.5.6 Deleting a Login Policy

Delete any custom login policy that is not assigned to a user.

Prerequisites

You must have the MANAGE ANY LOGIN POLICY system privilege.

You cannot delete a login policy if it is still assigned to a user. You must assign users to another login policy before deleting a customized login policy.

The root login policy cannot be deleted.

Context

The root login policy cannot be dropped, but it can be edited.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *Login Policies*.
3. On the *Login Policies* pane, right-click a login policy and click *Delete*.
4. Click *Yes*.

Results

The login policy is deleted.

Related Information

DROP LOGIN POLICY Statement

1.5.5.7 Login Policy Management for Read-only Databases

When you start a database in read-only mode, the login policies are based on the existing persistent state of the database.

The effect of any login policies you assign is limited to the current session.

If login management is enabled on a database that you later start in read-only mode, the following restrictions apply:

- Login management by the server is based on the state of the database before it is started.
- Explicit statements that change the state of the database are denied and result in an error.
- The database server continues to maintain dynamic information, such as failed_login_attempts and last_login_time, for each user. However, this information is maintained in transient memory and is lost when you shut down the database. The database returns to the same state it was in before you started it.
- If the account is locked by the existing login management policy, a user cannot log in. Also, the usual methods for changing a password during log in are unavailable.
- If the database is read-only because of its role as a mirror database in a high availability system or as a copy node in a read-only scale-out system, then the effect of any statement executed on the primary database is reflected in the read-only database. Also, the dynamic information collected on the primary server is sent to the read-only database and is merged in transient memory with the information collected for the read-only database.

1.5.5.8 Automatic Unlocking of User Accounts

A user account is automatically locked if the user exceeds the maximum failed login attempts limit (max_failed_login_attempts) value defined in the login policy.

Once locked, the user account can be manually unlocked using the ALTER USER ... RESET LOGIN POLICY statement by a user granted the MANAGE ANY USER system privilege. However, if all users with the MANAGE ANY USER system privilege are themselves locked out due to failed login attempts, a potential lock-down of some or all the services of a database can occur.

Automatic unlocking is applicable only to locked accounts due to failed login attempts and not to accounts locked for any other reason. The locked status of a user is verified during login, and if the user has equaled or exceeded the specified automatic unlock period, the user is allowed to log in and the failed_login_attempts counter is reset to zero.

Based on the permissions granted a user, one of the two login policy options is verified at the time of unlocking.

A lock-down of some or all of the services of a database can occur if all administrative users with the MANAGE ANY USER system privilege are locked out of the database due to failed login attempts.

To prevent this scenario, two login policy options are available:

root_auto_lock_time

Defines automatic unlocking period for users with the MANAGE ANY USER system privilege. Set this option to a small value (for example, 15 minutes). There is a server imposed upper limit of a few hours on this value. This option can be set in the root login policy only.

auto_unlock_time

Defines the automatic unlocking period for all other users. The default period is UNLIMITED, which means the user account will not be automatically unlocked. To enable automatic unlocking, the auto_unlock_time option can be set to a reasonable value (for example, 60 for a one-hour delay before automatic unlock occurs). The option can be set in any login policy, including the root login policy.

Configuration of these values requires the MANAGE ANY LOGIN POLICY system privilege.

Related Information

Login Policy Options and Default Values [page 642]

1.5.6 Database Options

Set database options by using the SET OPTION statement.

The syntax for this statement is as follows:

```
SET [ EXISTING ] [ TEMPORARY ] OPTION [ userid. | PUBLIC. ]option-name =
[ option-value ]
```

You can specify a user ID to set the option for that user. You can also specify the PUBLIC role name, which sets the option for all users that have not had an option explicitly set for them. If no user ID or PUBLIC is specified,

the option change is applied to the currently logged in user that issued the SET OPTION statement. The following are examples of setting a value for a PUBLIC option and then setting a different value for the user Browser.

```
SET OPTION PUBLIC.isolation_level = 1;
SET OPTION Browser.isolation_level = 2;
```

A user-defined option can also be set, however the option must have a PUBLIC setting before a user-specific value can be assigned. The database server does not support setting TEMPORARY values for user-defined options. The following example creates a user-defined option and sets it to a different value for the DBA user.

```
SET OPTION PUBLIC.user_defined_option = 'allusers';
SET OPTION DBA.user_defined_option = 'DBAonly';
```

You can set database options at two levels of scope:

- roles
- users

Setting an option for a role is only meaningful for the PUBLIC role and user-extended roles. Setting an option for a user-extended role only applies to that user when they are logged in. A user who inherits a role does not inherit that role's option settings.

The order of precedence for option settings is as follows:

1. All users inherit the PUBLIC role option setting unless the option is set specifically for the user.
2. A temporary PUBLIC option setting takes precedence over a permanent PUBLIC option setting.
3. A user option setting takes precedence over a temporary or permanent PUBLIC role setting.
4. A temporary user option setting for takes precedence over a permanent user option setting, a temporary PUBLIC option setting, and a permanent PUBLIC option setting. A temporary user option can only be set for the currently connected user.

A permanent setting is recorded in the database to which the user issuing a SET OPTION statement is connected.

If you set a permanent option for a user, the corresponding temporary option is set for the user as well. If you set a permanent PUBLIC option, the corresponding temporary PUBLIC option is set as well.

A temporary PUBLIC option setting applies to the current connection, and all new connections. In general, setting a temporary PUBLIC option does not apply to other existing connections unless otherwise noted in the description of the option.

A temporary user option setting applies to the current connection only.

When the option-value is omitted, the specified option is removed for the specified user.

When a temporary or permanent option setting is removed then the option setting reverts to the next temporary or permanent setting according to the order of precedence presented earlier. The following example will help to illustrate the hierarchy. Suppose that the DBA user executes the following statements.

```
SET OPTION PUBLIC.isolation_level = 0;
SET TEMPORARY OPTION PUBLIC.isolation_level = 1;
SET OPTION isolation_level = 2;
SET TEMPORARY OPTION isolation_level = 3;
SELECT connection_property( 'isolation_level' );
```

The isolation level for the current user (DBA) is 3. Other users who log in as DBA, have an isolation level of 2 because the isolation level has been set permanently for them. Users other than DBA have a temporary isolation level of 1 until the database is shut down and restarted (assuming that the isolation level has not been set permanently for them).

When the temporary option setting for the current user (DBA) is removed, the setting reverts to the permanent user setting for DBA. This is shown next.

```
SET TEMPORARY OPTION isolation_level =;
SELECT connection_property( 'isolation_level' );
```

The isolation level for the current user (DBA) is now 2. When the permanent option setting for the user DBA is removed, the setting reverts to the temporary PUBLIC setting. This is shown next.

```
SET OPTION isolation_level =;
SELECT connection_property( 'isolation_level' );
```

The isolation level for the current user (DBA) is now 1. When a temporary option setting for PUBLIC is removed, the setting reverts to the permanent PUBLIC setting. This is shown next.

```
SET TEMPORARY OPTION PUBLIC.isolation_level =;
SELECT connection_property( 'isolation_level' );
```

The isolation level for the current user (DBA) is now 0.

Some options (such as those that affect COMMIT behavior) are database-wide in scope. Setting these options requires specific privileges. Other options, such as the isolation_level option, can be applied to just the current connection and need no special privileges.

Changes to option settings take effect at different times, depending on the option. Changing a global database option such as the recovery_time option takes place the next time the database is started. Changing another user's database option takes place the next time the other user reconnects to the database. Generally, only options that affect the current user connection occur immediately. For example, you can change option settings in the middle of a transaction.

Some options that can only be set for the PUBLIC role take effect immediately for existing connections, even though the changed setting is not visible to users via the CONNECTION_PROPERTY function. An example of this behavior is the global_database_id option. For this reason, PUBLIC-only options should not be changed while other users are connected to the database.

⚠ Caution

Do not change option values while a cursor is open. Changing the option values while a cursor is open can lead to inconsistent results within the cursor. For example, changing the date_format option while a cursor is open can result in some rows being returned in the old format and some rows returned in the new format. To ensure that the rows in the result set are computed consistently using the new option value, open the cursor after the option value is changed.

> **i Note**
>
> In databases that use a Turkish collation or are case-sensitive, executing a query on the SYSOPTION system view or a query similar to the following might not match any rows if the option name is used with the wrong case:
>
> ```
> SELECT * FROM sa_conn_properties( ) WHERE propname = 'BLOCKING';
> ```

Example

The following statement applies an option change for the currently logged in user, in the database to which the user is currently connected.

```
SET OPTION blocking_timeout = 3;
```

The following statement applies an option change for the currently logged in user, for the duration of the connection only.

```
SET TEMPORARY OPTION blocking_timeout = 3;
```

The following statement applies an option change for the user Browser, in the database to which the user issuing the SET OPTION statement is currently connected.

```
SET OPTION Browser.blocking_timeout = 3;
```

The following statement applies a change to the PUBLIC role, in the database to which the user issuing the SET OPTION statement is currently connected. For the option in this example, the user issuing the SET OPTION statement must have the SET ANY SECURITY OPTION system privilege.

```
SET OPTION PUBLIC.login_mode = 'Standard';
```

The following statement removes the ansi_blanks option for the currently logged in user, in the database to which the user is currently connected. The ansi_blanks option for the currently logged in user reverts to the temporary or permanent PUBLIC role setting.

```
SET OPTION ansi_blanks =;
```

In this section:

Monitoring Option Settings [page 660]
Monitor option settings by using the sa_server_option system procedure with the either OptionWatchList or OptionWatchAction property to specify database server behavior when particular database option values are changed.

Changing Database Options (SQL Central) [page 661]
Change the values of database options for a database, user, or role.

Database Option Settings for TDS and Command Sequence Communication Protocols [page 663]
Connections to the database can be made through the TDS protocol (SAP Open Client and jConnect JDBC connections) or through the Command Sequence protocol (ODBC and Embedded SQL).

Alphabetical List of Database Options [page 663]
There are many database options you can set using the SET OPTION statement..

Different Categories of Database Options [page 873]
There are four categories of database options: compatibility, synchronization, SQL Remote, and Interactive SQL.

Related Information

SET OPTION Statement

1.5.6.1 Considerations for Setting Temporary Options

You can set temporary options by adding the TEMPORARY keyword to the SET OPTION statement changes the duration of the change.

Ordinarily, changing the value of an option is permanent.

When the SET TEMPORARY OPTION statement is used to set an option for the current user, the new option value takes effect only for the current connection, and only for the duration of the connection. The temporary setting is held only in server memory and is not reflected in the SYSOPTION system view.

When the SET TEMPORARY OPTION statement is used to set a PUBLIC option, the new option value takes effect only for the current connection and new connections that do not have a private user setting. If does not affect other existing connections. The temporary option value continues in effect until the database is shut down. The temporary setting is held only in server memory and is not reflected in the SYSOPTION system view.

Setting a temporary option for the PUBLIC role offers a security advantage. For example, when the login_mode option is enabled, the database relies on the login security of the system on which it is running. Enabling it as a temporary option setting means that a database relying on the security of a Windows domain is not compromised if the database is shut down and copied to a local computer. In this case, the login_mode option reverts to its permanent value, which could be Standard, a mode where Integrated user authentication is not permitted.

1.5.6.2 OPTION Clause in SQL Statements

The INSERT, UPDATE, DELETE, SELECT, UNION, EXCEPT, and INTERSECT statements have an OPTION clause that you can use to specify a database option setting that takes precedence over any PUBLIC or temporary option settings that are in effect, for that statement only.

You can also use the OPTION clause to specify how materialized views are used by the statement and how the query is optimized.

You can change the setting of the following database options in the OPTION clause:

- isolation_level
- max_query_tasks
- optimization_goal
- optimization_level
- optimization_workload
- user_estimates

Related Information

INSERT Statement
UPDATE Statement
DELETE Statement
SELECT Statement
UNION Statement
EXCEPT Statement
INTERSECT Statement
isolation_level Option [page 746]
max_query_tasks Option [page 767]
optimization_goal Option [page 785]
optimization_level Option [page 786]
optimization_workload Option [page 788]
user_estimates Option [page 862]

1.5.6.3 How to View Database Options

You can view system-defined, and user-defined database options using system procedures.

System-Defined Option Values

You can obtain a list of all system-defined option values, for your own connection, by using one of the following methods. To obtain system-defined option values for other connections, you must have either the SERVER OPERATOR, MONITOR, or DROP CONNECTION system privilege.

sa_conn_options system procedure

To list all current system-defined option settings, call the sa_conn_options system procedure:

```
CALL sa_conn_options;
```

To order this list alphabetically, execute the following statement:

```
SELECT *
FROM sa_conn_options( )
ORDER BY OptionName;
```

To filter the result or order by anything other than name, use a WHERE clause:

```
SELECT *
FROM sa_conn_options( )
WHERE OptionDescription LIKE '%date%'
ORDER BY PropNum;
```

sa_conn_properties system procedure

Current system-defined option settings for your connection are also available as a subset of **connection properties**. List all connection properties by using the sa_conn_properties system procedure:

```
CALL sa_conn_properties;
```

To order this list alphabetically, execute the following statement:

```
SELECT *
FROM sa_conn_properties( )
ORDER BY PropName;
```

SET statement

In Interactive SQL, execute the SET statement with no arguments to list the current settings of system-defined connection, database, and Interactive SQL options.

```
SET;
```

You can obtain the current value of a single system-defined option by using the CONNECTION_PROPERTY system function:

CONNECTION_PROPERTY system function

Execute the following statement to report the value of the ansi_blanks option:

```
SELECT CONNECTION_PROPERTY ( 'ansi_blanks' );
```

The CONNECTION_PROPERTY system function cannot be used to obtain the values of user-defined options.

Within Embedded SQL programs, you can use the GET OPTION statement to determine the value of an option (system- or user-defined) within your Embedded SQL application.

System-Defined Options That Can Only Be Set for the PUBLIC Role

Some options can only be set for the PUBLIC role and never for a user or connection. When you set one of these database options, the change may take effect immediately for existing connections, even though the changed setting is not visible to existing connections via the CONNECTION_PROPERTY function. In some cases, the change does not take effect immediately for existing connections and a disconnect/reconnect is required.

To view the current value for any PUBLIC option, query the SYSOPTION or SYSOPTIONS system views for the option name narrowing the search to the PUBLIC user using a WHERE clause. The difference between the SYSOPTION and SYSOPTIONS views is that the former contains the numeric user ID and the latter contains the user name.

Using the SYSOPTION view, the following query returns the PUBLIC setting for the specified option.

```
SELECT * FROM SYS.SYSOPTION
WHERE [option] = 'allow_snapshot_isolation'
AND [user_id] = (SELECT [user_ID] FROM SYS.SYSUSER WHERE [user_name] = 'PUBLIC');
```

Using the SYSOPTIONS view, the following query returns the PUBLIC setting for the specified option.

```
SELECT * FROM SYS.SYSOPTIONS
WHERE [option] = 'allow_snapshot_isolation'
AND [user_name] = 'PUBLIC';
```

User-Defined Option Values

You can obtain the value of user-defined options by querying the SYSOPTION or SYSOPTIONS system views.

A user-defined option must first be defined with a PUBLIC setting. Then you can define per-user settings. The following is an example.

```
SET OPTION PUBLIC.special = 'off';
SET OPTION DBA.special = 'on';
```

The values of user-defined options are not returned by either the sa_conn_options or sa_conn_properties system procedures. Instead , you can query the SYSOPTION or SYSOPTIONS catalog views directly to

determine the value of a specific user-defined option. The following query on the SYSOPTIONS system view displays all PUBLIC-level and user-level settings for the specified user-defined option:

```
SELECT *
FROM SYS.SYSOPTIONS
WHERE [option] = 'special';
```

The results of the sa_conn_options and sa_conn_properties system procedures do not include results for user-defined options.

All Option Values

To view the current value for all options, query the SYSOPTION or SYSOPTIONS system views. The difference between the SYSOPTION and SYSOPTIONS views is that the former contains the numeric user ID and the latter contains the user name.

```
SELECT *
FROM SYS.SYSOPTIONS;
```

You can also obtain the value of all options, except for temporary options, by using SQL Central:

SQL Central

In SQL Central, select a database, and then click *File* > *Options*. Make sure *Show: All Options* is selected.

Temporarily setting a server-defined option for a specific connection using the SET TEMPORARY OPTION statement does not cause the SYSOPTION or SYSOPTIONS system view to be updated. Temporary changes to option settings at the connection level are held only in server memory.

Related Information

sa_conn_options System Procedure
sa_conn_properties System Procedure
SYSOPTION System View
CONNECTION_PROPERTY Function [System]
GET OPTION Statement [ESQL]
SET OPTION Statement [Interactive SQL]
SET Statement [T-SQL]
SET REMOTE OPTION Statement [SQL Remote]

1.5.6.4 Monitoring Option Settings

Monitor option settings by using the sa_server_option system procedure with the either OptionWatchList or OptionWatchAction property to specify database server behavior when particular database option values are changed.

Prerequisites

You must have the SERVER OPERATOR system privilege.

Procedure

1. Connect to the database.
2. Call the sa_server_option system procedure with either the OptionWatchList or OptionWatchAction property.

   Use the sa_server_option system procedure to instruct the database server to send a message or return an error when an attempt is made to set a database option. Use the OptionWatchList property to create a list of options to monitor, and the OptionWatchAction property to specify the action the database server should take when an attempt is made to set an option that is being monitored.

Results

The option values are monitored.

Example

Call the following stored procedure to instruct the database server to monitor the database options automatic_timestamp, float_as_double, and tsql_hex_constant:

```
CALL dbo.sa_server_option(
'OptionWatchList','automatic_timestamp,float_as_double,tsql_hex_constant' );
```

Call the following stored procedure to instruct the database server to return an error if an attempt is made to set an option specified in the OptionWatchList property:

```
CALL dbo.sa_server_option( 'OptionWatchAction','ERROR' );
```

Related Information

sa_server_option System Procedure

1.5.6.5 Changing Database Options (SQL Central)

Change the values of database options for a database, user, or role.

Prerequisites

Any user can set their own options.

To set database options for other users or roles, including the PUBLIC role, you must have one of the following system privileges, depending upon which privilege the option requires:

- SET ANY SYSTEM OPTION
- SET ANY PUBLIC OPTION
- SET ANY SECURITY OPTION
- SET ANY USER DEFINED OPTION

Context

Changing the options for the database is equivalent to changing the options for the PUBLIC role.

Any option, whether user-defined or not, must have a PUBLIC setting before a user-specific value can be assigned. You can only create new options for the PUBLIC role.

Changing the value of an option for the PUBLIC role changes the value of the option for all users, except those who have explicitly set values for that option. When you set the value of an option for a user or role (other than the PUBLIC role), the new value overrides the value set by the PUBLIC role.

When you remove an option from a user or role, the PUBLIC setting for the option is applied. When you remove an option from the PUBLIC role, the option is removed from the database. All user settings for options must be removed before the option can be removed from the PUBLIC role.

You cannot set or change temporary options using SQL Central. To set a temporary option, use the SET OPTION statement.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.

2. Choose one of the following options:

Option	Action
Change the options for the PUBLIC role	1. In the left pane, click *Roles*. 2. In the right pane, right-click *PUBLIC* and click *Options*.
Change the options for a role	1. In the left pane, click *Roles*. 2. In the right pane, right-click the role and click *Options*. **i Note** Setting an option for a role is only meaningful for user-extended roles, and only applies to that user when they are logged in. A user who inherits a role does not inherit option settings.
Change the options for a user	1. In the left pane, click *Users*. 2. In the right pane, right-click the user and click *Options*.

3. Configure options for the user or role. The possible values you can specify appear in the *Value* field. Choose one of the following options:

Option	Action
Add an option	Click *New* and specify the option name and value.
Alter an option	Change the option value in the *Value* field, and then click *Set Permanent Now*.
Remove an option	Select the option and click *Remove Now*.

4. Click *Close*.

Results

The values of database options for the database, user, or role are changed.

Related Information

Alphabetical List of Database Options [page 663]
SET OPTION Statement

1.5.6.6 Database Option Settings for TDS and Command Sequence Communication Protocols

Connections to the database can be made through the TDS protocol (SAP Open Client and jConnect JDBC connections) or through the Command Sequence protocol (ODBC and Embedded SQL).

If you have users who use both TDS and Command Sequence (CmdSeq), you can configure their initial settings by using stored procedures. SQL Anywhere uses this method to set Open Client and jConnect connections to reflect default SAP Adaptive Server Enterprise behavior.

The initial settings are controlled by using the login_procedure option. This option names a stored procedure to run when users connect. The default setting is to use the sp_login_environment system procedure. You can change this behavior as necessary.

In turn, sp_login_environment checks to see if the connection is being made over TDS. If it is, it calls the sp_tsql_environment procedure, which sets several options to new default values for the current connection.

Related Information

Interface Library Communication Protocols
login_procedure Option [page 755]
sp_login_environment System Procedure
sp_tsql_environment System Procedure

1.5.6.7 Alphabetical List of Database Options

There are many database options you can set using the SET OPTION statement..

In this section:

allow_nulls_by_default Option [page 672]
Controls whether new columns that are created without specifying either NULL or NOT NULL are allowed to contain NULL values.

allow_optional_semicolon_in_tsql Option [page 673]
Controls the parsing of optional TSQL semicolons.

allow_read_client_file Option [page 674]
Controls whether to allow the reading of files on a client computer.

allow_snapshot_isolation Option [page 675]
Controls whether snapshot isolation is enabled or disabled.

allow_write_client_file Option [page 676]
Controls whether to allow the writing of files to a client computer.

ansi_blanks Option [page 677]
Controls behavior when character data is truncated at the client side.

ansi_close_cursors_on_rollback Option [page 679]

Controls whether cursors that were opened WITH HOLD are closed when a ROLLBACK is performed.

ansi_permissions Option [page 680]
Controls privileges checking for DELETE and UPDATE statements.

ansi_substring Option [page 681]
Controls the behavior of the SUBSTRING (SUBSTR) function when negative values are provided for the start or length parameters.

ansi_update_constraints Option [page 683]
Controls the range of updates that are permitted.

ansinull Option [page 684]
Controls the interpretation of NULL values.

audit_log Option [page 686]
Specifies the type and location of the audit log.

auditing Option [page 688]
Enables and disables auditing in the database.

auditing_options Option (Reserved for System Use) [page 690]
Reserved for system use. This database option is set by the sa_enable_auditing_type and sa_disable_auditing_type system procedures.

auto_commit Option [page 690]
Causes an automatic commit after every request.

auto_commit_on_create_local_temp_index Option [page 691]
Controls whether the database server performs a COMMIT before an index is created on a local temporary table.

background_priority Option [Deprecated] [page 692]
Limits impact on the performance of connections other than the current connection.

blob_threshold Option [SQL Remote] [page 693]
Controls the size of value that the Message Agent treats as a long object (BLOB).

blocking Option [page 694]
Controls the behavior in response to locking conflicts.

blocking_others_timeout Option [page 695]
Specifies the amount of time that another connection can block on the current connection's row and table locks before the current connection is rolled back.

blocking_timeout Option [page 697]
Controls how long a transaction waits to obtain a lock.

cache_estimated_value_plans Option [page 698]
Controls whether cached plans are optimized using value estimates.

chained Option [page 699]
Controls the transaction mode in the absence of a BEGIN TRANSACTION statement.

checkpoint_time Option [page 700]
Sets the maximum number of minutes that the database server runs without doing a checkpoint.

cis_option Option [page 701]
Controls whether debugging information for remote data access appears in the database server messages window.

Specifies the number of microseconds to add to a column that has a default value of TIMESTAMP or UTC TIMESTAMP to keep values in the column unique when a row containing the column is inserted or updated.

delayed_commit_timeout Option [page 726]
Specifies the maximum delay between an application executing a COMMIT and the COMMIT actually being written to disk when the delayed_commits option is set to On.

delayed_commits Option [page 727]
Determines when the database server returns control to an application following a COMMIT.

delete_old_logs Option [MobiLink][SQL Remote] [page 729]
Controls whether transaction logs are deleted when their transactions have been replicated or synchronized.

disk_sandbox Option [page 730]
Controls whether the read-write file operations of the database are restricted to the directory where the main database file is located and any subdirectories of this directory.

divide_by_zero_error Option [page 731]
Controls the reporting of division by zero.

escape_character Option (Reserved for System Use) [page 732]
Reserved for system use.

exclude_operators Option (Reserved for System Use) [page 733]
Reserved for system use. Do not change the setting of this option.

extended_join_syntax Option [page 733]
Controls whether queries with duplicate correlation names syntax for multi-table joins are allowed, or reported as an error.

extern_login_credentials Option [page 734]
Controls whether the external login credentials of the session user or the executing (effective) user are used when making remote connections. This option is provided for backwards compatibility. For security reasons, do not specify this option.

external_remote_options Option [SQL Remote] [page 735]
Indicates where the message link parameters should be stored.

fire_triggers Option [page 736]
Controls whether triggers are fired in the database.

first_day_of_week Option [page 737]
Sets the numbering of the days of the week.

for_xml_null_treatment Option [page 739]
Controls the treatment of NULL values in queries that use the FOR XML clause.

force_view_creation Option (Reserved for System Use) [page 740]
Reserved for system use. Do not change the setting of this option.

global_database_id Option [page 740]
Controls the range of values for columns created with DEFAULT GLOBAL AUTOINCREMENT.

http_connection_pool_basesize Option [page 741]
Specifies the nominal threshold size of database connections.

http_connection_pool_timeout Option [page 742]
Specifies the maximum duration that an unused connection can be retained in the connection pool.

Limits the maximum number of iterations a recursive common table expression can make.

max_statement_count Option [page 769]
Controls a resource governor that limits the maximum number of prepared statements that a connection can use simultaneously.

max_temp_space Option [page 770]
Controls the maximum amount of temporary file space a connection can use.

min_password_length Option [page 772]
Sets the minimum length for new passwords in the database.

min_role_admins Option [page 773]
Sets the minimum number of administrators required to administer roles.

ml_remote_id Option [MobiLink] [page 774]
Sets the remote ID for a remote database in a MobiLink synchronization system.

nearest_century Option [page 776]
Controls the interpretation of two-digit years in string-to-date conversions.

non_keywords Option [page 777]
Disables individual reserved keywords, allowing their use as identifiers.

odbc_describe_binary_as_varbinary Option [page 778]
Controls how the SQL Anywhere ODBC driver describes BINARY columns.

odbc_distinguish_char_and_varchar Option [page 779]
Controls how the SQL Anywhere ODBC driver describes CHAR columns.

oem_string Option [page 780]
Stores user-specified information in the header page of the database file.

on_charset_conversion_failure Option [page 782]
Controls what happens if an error is encountered during character conversion.

on_tsql_error Option [page 784]
Controls error-handling in stored procedures.

optimization_goal Option [page 785]
Determines whether query processing is optimized towards returning the first row quickly, or minimizing the cost of returning the complete result set.

optimization_level Option [page 786]
Controls the amount of effort made by the query optimizer to find an access plan for a SQL statement.

optimization_workload Option [page 788]
Determines whether query processing is optimized towards a workload that is a mix of updates and reads or a workload that is predominantly read-based (OLAP).

parameterization_level Option [page 789]
Controls automatic parameterization of client statements.

pinned_cursor_percent_of_cache Option [page 790]
Specifies how much of the cache can be used for pinning cursors.

post_login_procedure Option [page 791]
Specifies a procedure whose result set contains messages that should be displayed by applications when a user connects.

precision Option [page 793]

Specifies the maximum number of digits in the result of any decimal arithmetic.

prefetch Option [page 794]
Controls whether rows are fetched to the client side before being made available to the client application.

preserve_source_format Option [page 796]
Controls whether the original source definition of procedures, triggers, views, and event handlers is saved in system files.

prevent_article_pkey_update Option [MobiLink] [page 797]
Controls updates to the primary key columns of tables involved in MobiLink publications.

priority Option [page 798]
Sets the priority level at which requests from a connection are executed.

progress_messages Option [page 799]
Controls whether progress messages are sent from the database server to the client.

qualify_owners Option [SQL Remote] [page 801]
Controls whether SQL statements being replicated by SQL Remote should use qualified object names.

query_mem_timeout Option [page 802]
Sets the maximum time, in milliseconds, that a request waits for a memory grant.

quote_all_identifiers Option [SQL Remote] [page 803]
Controls whether SQL statements being replicated by SQL Remote should use quoted identifiers.

quoted_identifier Option [page 804]
Controls the interpretation of strings that are enclosed in double quotes.

read_past_deleted Option [page 805]
Controls database server behavior on uncommitted deletes at isolation levels 1 and 2.

recovery_time Option [page 806]
Sets the maximum length of time, in minutes, that the database server takes to recover from system failure.

remote_idle_timeout Option [page 807]
Controls how many seconds of inactivity web service client procedures and functions tolerate.

replication_error Option [SQL Remote] [page 808]
Allows you to specify a stored procedure to be called by the Message Agent when a SQL error occurs.

replication_error_piece Option [SQL Remote] [page 809]
Works with the replication_error option to allow you to specify a LONG VARCHAR stored procedure to be called by the Message Agent when a SQL error occurs during SQL Remote replication.

request_timeout Option [page 810]
Controls the maximum time a single request can run. This option can be used to prevent a connection from consuming a significant amount of server resources for a long period of time.

RESERVED_CONNECTIONS Option [page 811]
Controls the minimum number of concurrent connections that a database must reserve for standard connections. This option is useful when your database server accepts HTTP/HTTPS connections.

reserved_keywords Option [page 813]
Changes one or more keywords into reserved keywords.

return_date_time_as_string Option [page 814]

Controls how a DATE, TIME, or TIMESTAMP value is passed to the client application when queried.

rollback_on_deadlock Option [page 815]
Controls how transactions are treated when a deadlock occurs.

row_counts Option [page 816]
Specifies whether the database always count the number of rows in a query when it is opened.

save_remote_passwords Option [SQL Remote] [page 817]
Saves the password that is entered in the message link.

scale Option [page 818]
Specifies the minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum precision.

secure_feature_key Option [Deprecated] [page 819]
Allows you to enable features for the connection that were secured using the -sf database option.

sort_collation Option [page 821]
Allows implicit use of the SORTKEY function on ORDER BY expressions.

sql_flagger_error_level Option [page 822]
Controls the response to any SQL that is not part of the specified standard.

sql_flagger_warning_level Option [page 824]
Controls the response to any SQL that is not part of the specified standard.

sr_date_format Option [SQL Remote] [page 825]
Sets the format for DATE values that are retrieved from the database.

sr_time_format Option [SQL Remote] [page 827]
Sets the format for TIME values that are retrieved from the database.

sr_timestamp_format Option [SQL Remote] [page 828]
Sets the format for TIMESTAMP values that are retrieved from the database.

sr_timestamp_with_time_zone_format Option [SQL Remote] [page 830]
Sets the format for TIMESTAMP WITH TIME ZONE values retrieved from the database.

st_geometry_asbinary_format Option [page 832]
Controls how spatial values are converted from a geometry to binary.

st_geometry_astext_format Option [page 833]
Controls how spatial values are converted from a geometry to text.

st_geometry_asxml_format Option [page 834]
Controls how spatial values are converted from a geometry to XML.

st_geometry_describe_type Option [page 836]
Controls how spatial values are described.

st_geometry_interpolation Option [page 837]
Controls interpolation when spatial calculations involve circular arcs.

st_geometry_on_invalid Option [page 838]
Controls the behavior when a geometry fails basic validation (for example, a linestring with one point, a polygon with a ring that is not closed, or a geometry that exceeds the bounds of the spatial reference system).

string_rtruncation Option [page 839]
Determines whether an error is raised when a string is truncated.

1.5.6.7.1 allow_nulls_by_default Option

Controls whether new columns that are created without specifying either NULL or NOT NULL are allowed to contain NULL values.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The allow_nulls_by_default option is included for Transact-SQL compatibility.

Related Information

Open Client/jConnect TDS Compatibility Options [page 874]
Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
Options for Transact-SQL Compatibility
Tables That Are Compatible with Transact-SQL
CREATE TABLE Statement
ALTER TABLE Statement
sp_tsql_environment System Procedure

1.5.6.7.2 allow_optional_semicolon_in_tsql Option

Controls the parsing of optional TSQL semicolons.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	No	No

Remarks

When this option is On, the behaviour from SA17 onwards is used. That is, semicolons are allowed as optional statement delimiters in the TSQL dialect, and some mixed dialect procedure definitions that were valid in SA16 will be treated as syntax errors. When this option is Off, the pre-SA17 behaviour is used. Semicolons are not allowed as optional statement delimiters in TSQL and any procedure definitions affected by this change will be accepted by the parser.

This option can be set at the public scope only.

1.5.6.7.3 allow_read_client_file Option

Controls whether to allow the reading of files on a client computer.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SECURITY OPTION	Yes, with SET ANY SECURITY OPTION	Yes, with SET ANY SECURITY OPTION
Allowed to set temporarily?	Yes, with SET ANY SECURITY OPTION	Yes (current connection only), with SET ANY SECURITY OPTION	No

Remarks

Enable this option to read from files on a client computer, for example, by using the READ_CLIENT_FILE function.

Reading client files is not supported by the **Tabular Data Stream** (**TDS**) protocol.

Related Information

Access to Data on Client Computers
Client-side Data Security
READ_CLIENT_FILE Function [String]
LOAD TABLE Statement
isql_allow_read_client_file Option [Interactive SQL] [page 1064]
allow_write_client_file Option [page 676]
isql_allow_write_client_file Option [Interactive SQL] [page 1065]

1.5.6.7.4 allow_snapshot_isolation Option

Controls whether snapshot isolation is enabled or disabled.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

This option controls whether snapshot isolation is enabled for the database. Once this option is set to On, the database server starts recording the original versions of updated rows in the temporary file if a transaction uses snapshot isolation.

If there are transactions in progress when the setting of the allow_snapshot_isolation option is changed, then the change does not take effect immediately. Any transactions that are running when the option setting is changed from Off to On must complete before snapshots can be used. When the setting of the option is changed from On to Off, any outstanding snapshots are allowed to complete before the database server stops collecting version information, and new snapshots are not initiated.

You can view the current snapshot isolation setting for a database by querying the value of the SnapshotIsolationState database property:

```
SELECT DB_PROPERTY ( 'SnapshotIsolationState' );
```

Example

The following statement enables snapshot isolation for a database:

```
SET OPTION PUBLIC.allow_snapshot_isolation = 'On';
```

Related Information

Snapshot Isolation
Isolation Levels and Consistency
How to Enable Snapshot Isolation
isolation_level Option [page 746]
updatable_statement_isolation Option [page 860]

1.5.6.7.5 allow_write_client_file Option

Controls whether to allow the writing of files to a client computer.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SECURITY OPTION	Yes, with SET ANY SECURITY OPTION	Yes, with SET ANY SECURITY OPTION
Allowed to set temporarily?	Yes, with SET ANY SECURITY OPTION	Yes (current connection only), with SET ANY SECURITY OPTION	No

Remarks

Enable this option to write files to a client computer, for example, by using the WRITE_CLIENT_FILE function.

Writing client files is not supported by the **Tabular Data Stream** (**TDS**) protocol.

Related Information

Access to Data on Client Computers
Client-side Data Security
WRITE_CLIENT_FILE Function [String]
UNLOAD Statement
isql_allow_write_client_file Option [Interactive SQL] [page 1065]
allow_read_client_file Option [page 674]
isql_allow_read_client_file Option [Interactive SQL] [page 1064]

1.5.6.7.6 ansi_blanks Option

Controls behavior when character data is truncated at the client side.

Allowed Values

On, Off

Default

Off

A temporary setting for the current user is established by SAP Open Client and jConnect TDS connections ('On').

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The ansi_blanks option has no effect unless the database ignores trailing blanks in string comparisons and pads strings that are fetched into character arrays. It forces a truncation error whenever a value of data type CHAR(N) is read into a C char[M] variable for values of N greater than or equal to M. With ansi_blanks set to Off, a truncation error occurs only when at least one non-blank character is truncated.

For Embedded SQL with the ansi_blanks option set to On, when you supply a value of data type DT_STRING, you must set the sqllen field to the length of the buffer containing the value (at least the length of the value plus space for the terminating null character). With ansi_blanks set to Off, the length is determined solely by the position of the NULL character. The value of the ansi_blanks option is determined when the connection is established. Changing the option once the connection has been made does not affect this sqllen Embedded SQL behavior.

When a database is blank padded, this option controls truncation warnings sent to the client if the expression being fetched is CHAR or NCHAR (not VARCHAR or NVARCHAR) and it is being fetched into a CHAR or NCHAR (not VARCHAR or NVARCHAR) host variable. If these conditions hold and the host variable is too small to hold the fetched expression once it is blank padded to the expression's maximum length, a truncation warning is raised and the indicator contains the minimum number of bytes required to hold the fetched expression if it is blank padded to its maximum length. If the expression is CHAR(N) or NCHAR(N), the indicator may be set to a value other than N to take into account character set translation of the value returned and character length semantics.

Related Information

Open Client/jConnect TDS Compatibility Options [page 874]
Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
sp_tsql_environment System Procedure

1.5.6.7.7 ansi_close_cursors_on_rollback Option

Controls whether cursors that were opened WITH HOLD are closed when a ROLLBACK is performed.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The draft SQL/3 standard requires all cursors be closed when a transaction is rolled back. By default, on a rollback the database server closes only those cursors that were opened without a WITH HOLD clause. This option allows you to force closure of all cursors.

The close_on_endtrans option overrides the ansi_close_cursors_on_rollback option.

Related Information

Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
Cursors and Transactions
close_on_endtrans Option [page 702]

1.5.6.7.8 ansi_permissions Option

Controls privileges checking for DELETE and UPDATE statements.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

With ansi_permissions set to On, the ANSI/ISO SQL Standard privilege requirements for DELETE and UPDATE statements are checked. The default value is Off in Adaptive Server Enterprise. The following table outlines the differences.

SQL statement	Privileges required with ansi_permissions off	Privileges required with ansi_permissions on
UPDATE	UPDATE privilege on the columns where values are being set	UPDATE privilege on the columns where values are being set SELECT privilege on all columns appearing in the WHERE clause SELECT privilege on all columns on the right side of the SET clause

SQL statement	Privileges required with ansi_permissions off	Privileges required with ansi_permissions on
DELETE	DELETE privilege on the table	DELETE privilege on the table SELECT privilege on all columns appearing in the WHERE clause

Related Information

Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
SET Statement [T-SQL]

1.5.6.7.9 ansi_substring Option

Controls the behavior of the SUBSTRING (SUBSTR) function when negative values are provided for the start or length parameters.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

When the ansi_substring option is set to On, the behavior of the SUBSTRING function corresponds to ANSI/ISO SQL Standard behavior. A negative or zero start offset is treated as if the string were padded on the left with non-characters, and gives an error if a negative length is provided.

When this option is set to Off, the behavior of the SUBSTRING function is the same as in previous releases of SQL Anywhere: a negative start offset means an offset from the end of the string, and a negative length means the desired substring ends length characters to the left of the starting offset. Also, using a start offset of 0 is equivalent to a start offset of 1.

The setting of this option does not affect the behavior of the BYTE_SUBSTR function. Avoid using non-positive start offsets or negative lengths with the SUBSTRING function. Where possible, use the LEFT or RIGHT functions instead.

Example

The following examples show the difference in the values returned by the SUBSTRING function based on the setting of the ansi_substring option.

```
SUBSTRING( 'abcdefgh',-2,4 );
  ansi_substring = Off ==> 'gh' // substring starts at second-last character
  ansi_substring = On  ==> 'a'  // takes the first 4 characters of
                                // ???abcdefgh and discards all ?
SUBSTRING( 'abcdefgh',4,-2 );
  ansi_substring = Off ==> 'cd'
  ansi_substring = On  ==> value -2 out of range for destination
SUBSTRING( 'abcdefgh',0,4 );
  ansi_substring = Off ==> 'abcd'
  ansi_substring = On  ==> 'abc'
```

Related Information

Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
SUBSTRING Function [String]
LEFT Function [String]
RIGHT Function [String]

1.5.6.7.10 ansi_update_constraints Option

Controls the range of updates that are permitted.

Allowed Values

Off, Cursors, Strict

Default

Cursors

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The software provides several extensions that allow updates that the ANSI/ISO SQL Standard does not permit. These extensions provide powerful, efficient mechanisms for performing updates. However, they may cause unexpected behavior. This behavior can produce anomalies such as lost updates if the user application is not designed to expect the behavior of these extensions.

The ansi_update_constraints option controls whether updates are restricted to those updates permitted by the ANSI/ISO SQL Standard.

If the option is set to Strict, the following updates are prevented:

- Updates of cursors containing JOINS
- Updates of columns that appear in an ORDER BY clause
- The FROM clause is not allowed in UPDATE statements

If the option is set to Cursors, these same restrictions are in place, but only for cursors. If a cursor is not opened with FOR UPDATE or FOR READ ONLY, the database server chooses updatability based on the ANSI/ISO SQL Standard. If the ansi_update_constraints option is set to Cursors or Strict, cursors containing an ORDER BY clause default to FOR READ ONLY; otherwise, they continue to default to FOR UPDATE.

For ODBC, JDBC, ADO.NET, and OLE DB, statement updatability is explicitly READ ONLY.

For Embedded SQL, statement updatability is explicitly READ ONLY unless the SQL preprocessor -m HISTORICAL option is specified.

By default, stored procedure cursors are not explicitly FOR UPDATE or READ ONLY.

Related Information

Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
Row Updates and Deletes Through a Cursor
UPDATE Statement
The Embedded SQL Preprocessor

1.5.6.7.11 ansinull Option

Controls the interpretation of NULL values.

Allowed Values

On, Off

Default

On

A temporary setting for the current user is established by SAP Open Client and jConnect TDS connections ('Off').

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option is implemented primarily for Transact-SQL (Adaptive Server Enterprise) compatibility. The ansinull option affects the results of comparison predicates with NULL constants, and also affects warnings issued for grouped queries over NULL values.

With ansinull set to On, ANSI three-valued logic is used for all comparison predicates in a WHERE or HAVING clause, or in an On condition. Any comparisons with NULL using = or != evaluate to unknown.

Setting ansinull to Off means that the database server uses two-valued logic for the following four conditions:

`expr` = NULL

`expr` != NULL

`expr` = @var // @var is a procedure variable, or a host variable

`expr` != @var

In each case, the predicate evaluates to either true or false and never to unknown. In such comparisons, the NULL value is treated as a special value in each domain, and an equality (=) comparison of two NULL values yields true. The expression `expr` must be a relatively simple expression, referencing only columns, variables, and literals; subqueries and functions are not permitted.

With ansinull set to On, the evaluation of any aggregate function, except COUNT(*), on an expression that contains at least one NULL value, may generate a warning (SQLSTATE '01003').

With ansinull set to Off, this warning does not appear.

> **i** Note
>
> - Any SQL statement that contains a WHERE, HAVING, or ON clause is affected by the ansinull option. Also, any expression inside such a statement is affected by the ansinull option.
> - Adaptive Server Enterprise 12.5 introduced a change in the behavior of LIKE predicates with a NULL pattern string when ansinull is set to Off. In SQL Anywhere, LIKE predicates remain unaffected by the setting of ansinull.

Related Information

Open Client/jConnect TDS Compatibility Options [page 874]
Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
Null value eliminated in aggregate function
sp_tsql_environment System Procedure
SET Statement [T-SQL]

1.5.6.7.12 audit_log Option

Specifies the type and location of the audit log.

Allowed Values

```
[ TRANSLOG
  | SYSLOG
  | FILE ( filename_prefix='path-and-filename'; [ target-parameter [;...] ])
]
target-parameter :
  target-parameter-name = target-parameter-value[;...]
target-parameter-name
{
  | max_size;
  | num_files;
  | flush_on_write;
  | compressed;
}
```

Parameters

Parameter name	Value
filename_prefix	A log file name prefix with or without a path. All log files have the extension `.etd`. If a full path is not specified, then the directory where the database is located is used as the root directory. This parameter is required.
max_size	The maximum size of the file in bytes. The default is 0, which means there is no limit on the file size, and the file grows as long as disk space is available. Once the specified size is reached, a new file is started.
num_files	The number of files where event tracing information is written. This setting is used only if `max_size` is set. If all the files reach the maximum specified size, then the database server overwrites the oldest file.
flush_on_write	A Boolean value that controls whether disk buffers are flushed for each event that is logged. The default is ON. When this parameter is turned on, the performance of the database server may be reduced if many trace events are being logged.

Parameter name	Value
compressed	A Boolean value that controls compression of the log file to conserve disk space. The default is OFF.

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SECURITY OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SECURITY OPTION	No	No

Remarks

This option specifies where to write auditing events. If the audit_log option is not specified, or if an empty string is specified, then auditing events are logged to the transaction log (TRANSLOG) by default.

To turn off the transaction log while auditing is enabled, audit_log *must* be set to FILE or SYSLOG. If audit_log is set to the transaction log, then the transaction log cannot be turned off (until a file is specified).

If audit_log is set to SYSLOG, then events are logged to your operating system's event tracing log. The event tracing log differs between Microsoft Windows and UNIX/Linux:

- On Microsoft Windows operating systems, the event tracing log is the Microsoft *Windows Event* log. Events have source names beginning with SQLANY and can be viewed by navigating to *Event Viewer (Local)* > *Windows Logs* > *Applications* in the Event Viewer Utility.
- On UNIX and Linux operating systems, the event tracing log is the syslog facility.

Duplicate target names are not allowed. If FILE, TRANSLOG, or SYSLOG is specified twice, then an error is returned.

When disk sandboxing is enabled, any file referenced in the audit_log option must be in an accessible location.

If the value of audit_log is invalid, or if any audit log target cannot be started, then an error is returned and the previous value of audit_log is restored. If the previous value cannot be restored, then the audit_log option defaults to the transaction log, and a message appears in the database server messages window. However, the previous value of the audit_log option is maintained.

If an attempt to log in to the file fails, then audit events may be missing from a FILE target. If a logging failure occurs, then a message appears in the database server messages window to indicate the earliest occurrence of a failure.

For the audit_log option to work, set the auditing option to On, and also specify which types of information you want to audit by using the sa_enable_auditing_type system procedure.

If a FILE target is specified, then the database uses an internal trace event session named **audit** to log auditing events to the specified file. The internal trace event session cannot be modified by the user. The trace event session used for auditing differs from user trace event sessions in the following ways:

- If a full path is not specified for the filename_prefix parameter, then the directory where the database is located is used as the root directory.
- The flush_on_write parameter is set to ON by default.

Example

The following statement sets the audit log to the database's transaction log.

```
SET OPTION PUBLIC.audit_log = 'TRANSLOG'
```

The following statement sets the audit log to a file target.

```
SET OPTION PUBLIC.audit_log = 'FILE(filename_prefix=c:\audit)'
```

Related Information

Database Activity Audits [page 1692]
Viewing the Contents of the Event Trace Data (ETD) Diagnostic Log File [page 1018]
auditing Option [page 688]
sa_enable_auditing_type System Procedure
Event Trace Data (ETD) File Management Utility (dbmanageetd) [page 1157]

1.5.6.7.13 auditing Option

Enables and disables auditing in the database.

Allowed Values

ON, OFF

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SECURITY OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SECURITY OPTION	No	No

Remarks

This option turns auditing on and off.

Auditing is the recording of details about events in the database. Auditing provides some security features at a performance cost.

The following restrictions are in place when the audit log is set to the transaction log:

- When you turn on auditing for a database, you cannot stop using the transaction log. In this case, you must turn off auditing before you turn off the transaction log.
- Databases with auditing set to the transaction log cannot be started in read-only mode.

If you set the auditing option to On, and do not specify auditing options, then all types of auditing information are recorded. Alternatively, to control the type of information that you want to audit use the sa_enable_auditing_type and sa_disable_auditing_type system procedures.

Example

The following statement turns on auditing for the database.

```
SET OPTION PUBLIC.auditing = 'On';
```

Related Information

Database Activity Audits [page 1692]
audit_log Option [page 686]
sa_audit_string System Procedure

sa_enable_auditing_type System Procedure
sa_disable_auditing_type System Procedure

1.5.6.7.14 auditing_options Option (Reserved for System Use)

Reserved for system use. This database option is set by the sa_enable_auditing_type and sa_disable_auditing_type system procedures.

1.5.6.7.15 auto_commit Option

Causes an automatic commit after every request.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	No	No	No
Allowed to set temporarily?	No	Yes (current connection only)	No

Remarks

If this option is set to On, then the database server automatically commits after every request. This option can only be set temporarily for a connection.

When an application enables automatic commit using the specific driver API, the SQL Anywhere JDBC, ODBC, ADO.NET, and OLE DB drivers automatically set the auto_commit option to On if they are connected to a version 17 database server. For version 16 or earlier database servers, the driver reverts back to handling automatic commits on the client side. By default, automatic commit is enabled for these drivers.

> **Note**
>
> Do not set the auto_commit server option directly when using an API such as JDBC, ODBC, ADO.NET, or OLE DB. Use the API-specific mechanism for enabling or disabling automatic commit. For example, in ODBC set the SQL_ATTR_AUTOCOMMIT connection attribute using SQLSetConnectAttr. When you use the API, the driver can track the current setting of automatic commit.

> **Note**
>
> Use a BEGIN block to set the database option from an Interactive SQL session to avoid setting the Interactive SQL option of the same name:
>
> ```
> BEGIN
>     SET TEMPORARY OPTION AUTO_COMMIT = 'ON';
> END;
> ```
>
> Use this Interactive SQL command to verify the new setting of the database option:
>
> ```
> SET;
> ```

> **Note**
>
> The auto_commit option is different from the chained option. Setting auto_commit to On forces the database server to commit after every request. Setting the chained option to Off forces the database server to commit after each statement. This distinction is most important when executing a stored procedure. Setting the chained option to Off will result in a commit request after the execution of each individual statement within the procedure. Setting the auto_commit option to On will result in a single commit request once the entire procedure finishes executing. In cases where automatic commit is necessary, it is much better to use the auto_commit option rather than the chained option.

1.5.6.7.16 auto_commit_on_create_local_temp_index Option

Controls whether the database server performs a COMMIT before an index is created on a local temporary table.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

When this option is set to On, the database server executes a COMMIT before an index is created on a local temporary table. If the commit fails, the operation fails or rolls back. When this option is set to Off (the default), a COMMIT is not performed, which allows a procedure that relies on the index to be used as part of an atomic operation (such as a trigger, UPDATE statement, INSERT statement, DELETE statement, or the FROM clause).

Related Information

CREATE INDEX Statement

1.5.6.7.17 background_priority Option [Deprecated]

Limits impact on the performance of connections other than the current connection.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

When this option is set to On, requests execute at the Background priority level. When this option is set to Off, requests execute at the value specified by the Priority option.

Intra-query parallelism is not used for connections with background_priority set to On.

Related Information

priority Option [page 798]
max_priority Option [page 766]

1.5.6.7.18 blob_threshold Option [SQL Remote]

Controls the size of value that the Message Agent treats as a long object (BLOB).

Allowed Values

Integer, in bytes

Default

256

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

Any value longer than the blob_threshold option is replicated as a BLOB. That is, it is broken into pieces and replicated in chunks before being reconstituted using a SQL variable and concatenating the pieces at the recipient site.

Each SQL statement must fit within a message, so do not set the value of this option to a size larger than your message size (50 KB by default).

Related Information

SQL Remote Options
BLOBs

1.5.6.7.19 blocking Option

Controls the behavior in response to locking conflicts.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

If the blocking option is set to On, any transaction attempting to obtain a lock that conflicts with an existing lock held by another transaction waits until every conflicting lock is released or until the blocking_timeout is reached. If the lock is not released within blocking_timeout milliseconds, then an error is returned for the waiting transaction. If the blocking option is set to Off, the transaction that attempts to obtain a conflicting lock receives an error.

The value for the blocking connection property can be temporarily set to Off during certain operations (for example, refreshing a materialized view), which can then cause an error.

Related Information

The Blocking Option
blocking_timeout Option [page 697]

1.5.6.7.20 blocking_others_timeout Option

Specifies the amount of time that another connection can block on the current connection's row and table locks before the current connection is rolled back.

This option can be used to prevent a low priority task from blocking other connections for longer than the specified time.

Allowed Values

Integer, in milliseconds

Default

0

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	No	No	No
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

When this option is set to 0, other connections can block on the current connection for an indefinite period of time.

If another connection is blocked on the current connection for longer than the number of milliseconds specified by this option, the current connection is rolled back. If the current connection is in the middle of a request, the request is interrupted. An error is returned if the connection was rolled back. The error can be returned twice if a request must be interrupted. If the connection was idle, the connection is rolled back immediately. The application must be prepared to fail or retry after a delay if it is rolled back.

> **i** Note
>
> Locks created by LOCK TABLE `table-name` WITH HOLD are rolled back. Once a connection receives an error indicating that an operation was rolled back because of a blocking_others_timeout error, that connection receives the error on every request until the connection disconnects.

Related Information

Transaction Blocking and Deadlock
Automatically Release Schema Locks (Interactive SQL) [page 1085]
blocking_timeout Option [page 697]
LOAD TABLE Statement

1.5.6.7.21 blocking_timeout Option

Controls how long a transaction waits to obtain a lock.

Allowed Values

Integer, in milliseconds

Default

0

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

When the blocking option is set to On, any transaction attempting to obtain a lock that conflicts with an existing lock waits for blocking_timeout milliseconds for the conflicting lock to be released. If the lock is not released within blocking_timeout milliseconds, then an error is returned for the waiting transaction.

Setting this option to 0 forces all transactions attempting to obtain a lock to wait until all conflicting transactions release their locks.

Related Information

Deadlocks
blocking Option [page 694]
blocking_others_timeout Option [page 695]
request_timeout Option [page 810]

1.5.6.7.22 cache_estimated_value_plans Option

Controls whether cached plans are optimized using value estimates.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

If set to On, cached plans are optimized using value estimates influenced by the actual values provided with the statements when the plan was cached. If set to Off, cached plans are optimized independently of the actual values provided.

> **i** Note
>
> Off corresponds to the behavior in versions 16 and earlier of the software.

1.5.6.7.23 chained Option

Controls the transaction mode in the absence of a BEGIN TRANSACTION statement.

Allowed Values

On, Off

Default

On

A temporary setting for the current user is established by SAP Open Client and jConnect TDS connections ('Off').

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

Controls the Transact-SQL transaction mode. In Unchained mode (chained=Off), each statement is committed individually unless an explicit BEGIN TRANSACTION statement is executed to start a transaction. In chained mode (chained=On) a transaction is implicitly started before any data retrieval or modification statement.

Related Information

Open Client/jConnect TDS Compatibility Options [page 874]
Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
Autocommit Implementation Details
Transactions
sp_tsql_environment System Procedure

1.5.6.7.24 checkpoint_time Option

Sets the maximum number of minutes that the database server runs without doing a checkpoint.

Allowed Values

Integer

Default

60

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

This option is used with the recovery_time option to decide when checkpoints should be done.

You may need to shut down and restart the database server for the change to take effect.

Related Information

Checkpoint Logs [page 307]
How the Database Server Decides When to Checkpoint [page 306]
recovery_time Option [page 806]
-gc Database Server Option [page 429]
-m Database Server Option [page 465]
CHECKPOINT Statement

1.5.6.7.25 cis_option Option

Controls whether debugging information for remote data access appears in the database server messages window.

Allowed Values

0, 7

Default

0

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option controls whether information about how queries are executed on a remote database appears in the database server messages window when using remote data access. Set this option to 7 to see debugging information in the database server messages window. When this option is set to 0 (the default), debugging information for remote data access does not appear in the database server messages window.

Once you have turned on remote tracing, the tracing information appears in the database server messages window. You can log this output to a file by specifying the -o server option when you start the database server.

Related Information

-o Database Server Option [page 471]
Troubleshooting: Connectivity Tests for Remote Data Access

1.5.6.7.26 cis_rowset_size Option

Sets the number of rows that are returned from remote servers for each fetch.

Allowed Values

Integer

Default

50

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option takes effect when a new connection is made to a remote server.

This option sets the ODBC FetchArraySize value when using ODBC to connect to a remote database server.

1.5.6.7.27 close_on_endtrans Option

Controls the closing of cursors at the end of a transaction.

Allowed Values

On, Off

Default

On

A temporary setting for the current user is established by SAP Open Client and jConnect TDS connections ('Off').

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

When close_on_endtrans is set to On, cursors are closed whenever a transaction is committed unless the cursor was opened WITH HOLD. The behavior when a transaction is rolled back is governed by the ansi_close_cursors_on_rollback option.

When close_on_endtrans is set to Off, cursors are not closed at either a commit or a rollback, regardless of the ansi_close_cursors_on_rollback option setting or whether the cursor was opened WITH HOLD or not.

Setting this option to Off provides Adaptive Server Enterprise compatible behavior.

Related Information

Open Client/jConnect TDS Compatibility Options [page 874]
Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
ansi_close_cursors_on_rollback Option [page 679]
sp_tsql_environment System Procedure
SET Statement [T-SQL]

1.5.6.7.28 collect_statistics_on_dml_updates Option

Controls the gathering of statistics during the execution of data-altering DML statements such as INSERT, DELETE, and UPDATE.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The database server updates statistics during normal statement execution and uses the gathered statistics to self-tune column statistics. Set the collect_statistics_on_dml_updates option to Off to disable the updating of statistics during the execution of data-altering DML statements such as INSERT, DELETE, and UPDATE.

Under normal circumstances, it is not necessary to turn off this option. However, in environments where large amounts of data are frequently changing, setting this option to Off can improve performance when update_statistics is also set to On.

The difference between the collect_statistics_on_dml_updates option and the update_statistics option is that the update_statistics option compares the actual number of rows that satisfy a predicate with the number of rows that are estimated to satisfy the predicate, and then updates the estimates. The collect_statistics_on_dml_updates option modifies the column statistics based on the values of the specific rows that are inserted, updated, or deleted.

Related Information

update_statistics Option [page 861]

1.5.6.7.29 compression Option [SQL Remote]

Sets the level of compression for SQL Remote messages.

Allowed Values

Integer, from -1 to 9

Default

6

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

The values have the following meanings:

-1

Send messages in version 5 format. The Message Agent from version 5 cannot read messages sent by the Message Agent from version 6 and later. Ensure that the compression option is set to -1 until all Message Agents in your system are upgraded to version 6 or later.

0

No compression.

1 to 9

Increasing degrees of compression. Creating messages with high compression can take longer than creating messages with low compression.

Related Information

Message Size
SQL Remote Options

1.5.6.7.30 conn_auditing Option

Controls whether auditing is enabled or disabled for each connection when the auditing option is set to On.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	No	No	No
Allowed to set temporarily?	Yes with SET ANY SECURITY OPTION	Yes (current connection only)	No

Remarks

The setting of the conn_auditing option is only respected when it is set in a login procedure (specified by the login_procedure database option). Setting conn_auditing to On turns on auditing for the connection. However, auditing information is not recorded unless the auditing option is also set to On. To determine whether a connection is being audited, execute the following statement:

```
SELECT CONNECTION_PROPERTY ( 'conn_auditing' );
```

Related Information

auditing Option [page 688]
login_procedure Option [page 755]

1.5.6.7.31 connection_authentication Option

Specifies an authentication string that is used to verify the application signature against the database signature for authenticated applications.

Allowed Values

String

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	No	No	No
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option only takes effect when you are using the OEM Edition of the SQL Anywhere database server that supports OEM authentication.

Authenticated applications must set the connection_authentication database option for every connection immediately after the connection is established. If the signature is verified, the connection is authenticated and has no restrictions on its activities beyond those imposed by the SQL privileges. If the signature is not verified, the connection is limited to those actions permitted by unauthenticated applications.

Set the connection_authentication option for the duration of only the current connection by using the TEMPORARY keyword. The following SQL statement authenticates the connection:

```
SET TEMPORARY OPTION connection_authentication =
     'company = company-name;
     application = application-name;
     signature = application-signature';
```

The `company-name` and `application-name` must match the values specified by the database_authentication option. The `application-signature` is the application signature that you obtained from SAP.

If your company name has quotation marks, apostrophes, or other special characters, double them in the string for it to be accepted.

Example

The following example specifies an authentication string that contains special characters:

```
SET TEMPORARY OPTION connection_authentication=
   'Company = Joe''s Garage;
    Application = Joe''s Program;
    Signature = 0fa55157edb8e14d818e...';
```

Related Information

Authenticated Applications for OEM Editions [page 369]
database_authentication Option [page 715]

1.5.6.7.32 connection_type Database Option

Controls whether connections from monitoring applications affect database or database server shutdown.

Allowed Values

Event, Internal, Standard, Monitor

The default values are as follows:

Standard

For CmdSeq, TDS, HTTP, and HTTPS connections.

Event

For connections created for event handlers.

Internal

For all other types of connections.

Default

Standard

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	No	No	No
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

For CmdSeq, TDS, HTTP, and HTTPS connections, the supported values are **`Standard`** and **`Monitor`**.

Connections set to **`Monitor`** are ignored when determining whether the conditions have been met for shutting down the database or database server when using one of the following:

- The *Shut down* button on the database server messages window.
- The dbstop utility without the -y option.
- The STOP DATABASE or STOP SERVER statement without the UNCONDITIONALLY clause.

If a database is configured to automatically stop, connections with connection_type set to Monitor are ignored when determining whether the conditions have been met for automatically stopping the database.

> **i** Note
>
> HTTP and HTTPS connections *without* associated session IDs are always ignored when determining whether the conditions have been met for shutting down the database or database server, even if the connection type is not set to **`Monitor`**.
>
> HTTP and HTTPS connections *with* associated session IDs are only ignored when determining whether the conditions have been met for shutting down the database or database server if the connection type is set to **`Monitor`**.

Example

To ensure that the database server ignores your connection when determining whether the conditions have been met for shutting down the database or database server, execute the following statement:

```
SET TEMPORARY OPTION connection_type = 'Monitor';
```

1.5.6.7.33 continue_after_raiserror Option

Controls behavior following a RAISERROR statement.

Allowed Values

On, Off

Default

On

A temporary setting for the current user is established by SAP Open Client and jConnect TDS connections ('On').

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The RAISERROR statement is used within procedures and triggers to generate an error. When this option is set to Off, the execution of the procedure or trigger is stopped whenever the RAISERROR statement is encountered.

This option is ignored within the TRY block of a BEGIN...END statement.

If you set the continue_after_raiserror option to On, the RAISERROR statement no longer signals an execution-ending error. Instead, the RAISERROR status code and message are stored and the most recent RAISERROR is returned when the procedure completes. If the procedure that caused the RAISERROR was called from another procedure, the RAISERROR is not returned until the outermost calling procedure ends.

Intermediate RAISERROR statuses and codes are lost after the procedure ends. If, at return time, an error occurs along with the RAISERROR, then the information for the new error is returned and the RAISERROR information is lost. The application can query intermediate RAISERROR statuses by examining the @@error global variable at different execution points.

The setting of the continue_after_raiserror option is used to control behavior following a RAISERROR statement only if the on_tsql_error option is set to Conditional (the default). If you set the on_tsql_error option to Stop or Continue, the on_tsql_error setting takes precedence over the continue_after_raiserror setting.

Related Information

Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
on_tsql_error Option [page 784]
RAISERROR Statement
BEGIN Statement

1.5.6.7.34 conversion_error Option

Controls the reporting of data type conversion failures on fetching information from the database.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option controls whether data type conversion failures, when data is fetched from the database or inserted into the database, are reported by the database as errors (conversion_error set to On) or as a warning (conversion_error set to Off).

When conversion_error is set to On, the SQLE_CONVERSION_ERROR or SQLE_OVERFLOW_ERROR errors are generated. If the option is set to Off, the warning SQLE_CANNOT_CONVERT is produced.

If conversion errors are reported as warnings only, the NULL value is used in place of the value that could not be converted. In Embedded SQL, an indicator variable is set to -2 for the column or columns that cause the error.

Related Information

Indicator Variables: Conversion Errors
Conversion Errors During Import
Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]

1.5.6.7.35 cooperative_commit_timeout Option [Deprecated]

Governs when a COMMIT entry in the transaction log is written to disk.

> **i** Note
>
> As of version 11, this option setting is ignored as commit behavior is automatically tuned.

Allowed Values

Integer, in milliseconds

Default

250

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option has meaning only when cooperative_commits is set to On. The database server waits for the specified number of milliseconds for other connections to fill a page of the log before writing to disk. The default setting is 250 milliseconds.

Related Information

cooperative_commits Option [Deprecated] [page 713]
delayed_commit_timeout Option [page 726]
COMMIT Statement

1.5.6.7.36 cooperative_commits Option [Deprecated]

Controls when commits are written to disk.

> **i** Note
>
> As of version 11, this option setting is ignored as commit behavior is automatically tuned.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

If cooperative_commits is set to Off, a COMMIT is written to disk when the database server receives it, and the application is then allowed to continue.

If cooperative_commits is set to On (the default) and if there are other active connections, the database server does not immediately write the COMMIT to the disk. Instead, the application waits for up to the maximum length set by the cooperative_commit_timeout option for something else to put on the pages before they are written to disk.

Setting cooperative_commits to On, and increasing the cooperative_commit_timeout setting, increases overall database server throughput by cutting down the number of disk I/Os, but at the expense of a longer turnaround time for each individual connection.

If both cooperative_commits and delayed_commits are set to On, and the cooperative_commit_timeout interval passes without the pages getting written, the application is resumed (as if the commit had worked), and the remaining interval (delayed_commit_timeout - cooperative_commit_timeout) is used as a delayed_commits interval. The pages are then written, even if they are not full.

Related Information

delayed_commits Option [page 727]
COMMIT Statement

1.5.6.7.37 database_authentication Option

Sets the authentication string for a database.

Allowed Values

String

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SECURITY OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SECURITY OPTION	No	No

Remarks

This option only takes effect when you are using the OEM Edition of the SQL Anywhere database server that supports OEM authentication.

You may need to restart the database for this option to take effect.

When a database is authenticated, only connections that specify the correct authentication signature can perform operations on the database. Connections that are not authenticated operate in read-only mode. To use authenticated databases, use the OEM Edition of SQL Anywhere.

To authenticate a database, set the database_authentication option for the database:

```
SET OPTION PUBLIC.database_authentication =
 'company = company-name;
 application = application-name;
 signature = database-signature';
```

The `company-name` and `application-name` arguments are the values you supplied to SAP when obtaining your signature, and `database-signature` is the database signature that you received from SAP.

If your company name has quotation marks, apostrophes, or other special characters, double them in the string for it to be accepted.

When the database server loads an authenticated database, it displays a message in the database server messages window describing the authenticated company and application. You can check that this message is present to verify that the database_authentication option has taken effect. The message has the following form:

```
This database is licensed for use with:
Application: application-name
Company: company-name
```

You can store the authentication statement in a SQL script file to avoid having to type in the long signature repeatedly. If you store the authentication statement in the file *%SQLANY17%*`\scripts\authenticate.sql`, it is applied whenever you create, rebuild, or upgrade a database.

Example

```
SET OPTION PUBLIC.database_authentication =
     'company = MyCompany;
     application = MyApp;
     signature = 0fa55157edb8e14d818e';
```

Related Information

Authenticated Applications for OEM Editions [page 369]
Upgrading Authenticated Databases [page 377]
connection_authentication Option [page 707]

1.5.6.7.38 date_format Option

Sets the format for dates that are retrieved as strings from the database.

Allowed Values

String

Default

YYYY-MM-DD

A temporary setting for the current user is established by the ODBC, JDBC, and OLE DB drivers ('yyyy-mm-dd').

A temporary setting for the current user is established by SAP Open Client and jConnect TDS connections ('YYYY-MM-DD').

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The format is a string using the following symbols:

Symbol	Description
YY	Two-digit year
YYYY	Four-digit year
MM	Two-digit month
MMM[m...]	Character short form for months
D	Single digit day of week (1 = Sunday, 7 = Saturday)
DD	Two-digit day of month
DDD[d...]	Character short form for days of the week
JJJ	Day of the year, from 1 to 366

Each symbol is substituted with the appropriate data for the date that is being formatted.

If the character data is multibyte, the length of each symbol reflects the number of characters, not the number of bytes. For example, the 'mmm' symbol specifies a length of three characters for the month.

For symbols that represent character data (such as MMM), you can control the case of the output as follows:

- Type the symbol in all uppercase to have the format appear in all uppercase. For example, MMM produces JAN.
- Type the symbol in all lowercase to have the format appear in all lowercase. For example, mmm produces jan.

- Type the symbol in mixed case to have the database server choose the appropriate case for the language that is being used. For example, in English, typing Mmm produces May, while in French it produces mai.

For symbols that represent numeric data, you can control zero-padding with the case of the symbols:

- Type the symbol in same-case (such as MM or mm) to allow zero padding. For example, yyyy/mm/dd could produce 2002/01/01.
- Type the symbol in mixed case (such as Mm) to suppress zero padding. For example, yyyy/Mm/Dd could produce 2002/1/1.

> **i Note**
>
> If you change the setting for date_format in a way that re-orders the date format, change the date_order option to reflect the same change, and vice versa.

Example

The following table illustrates date_format settings, together with the output from the following statement, executed on Monday, April 14, 2008.

```
SELECT CAST( CURRENT DATE AS VARCHAR );
```

date_format	CURRENT DATE
yyyy/mm/dd/ddd	2008/04/14/mon
yyyy/Mm/Dd/ddd	2008/4/14/mon
jjj	105
mmm yyyy	apr 2008
Mmm yyyy	Apr 2008
mm-yyyy	04-2008

Related Information

Search Conditions That Compare Dates
Open Client/jConnect TDS Compatibility Options [page 874]
Server Options Changed by ODBC
date_order Option [page 719]
DATE Data Type
time_format Option [page 847]
timestamp_format Option [page 850]
timestamp_with_time_zone_format Option [page 852]
sp_tsql_environment System Procedure

1.5.6.7.39 date_order Option

Controls the interpretation of date formats.

Allowed Values

MDY, YMD, DMY

Default

YMD

A temporary setting for the current user is established by the ODBC, JDBC, and OLE DB drivers ('ymd').

A temporary setting for the current user is established by SAP Open Client and jConnect TDS connections ('MDY').

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The default order for year, month, and day corresponds to the ISO 8601 date format. For example, "06-01-07" is interpreted as January 7, 2006.

You can specify a different order for the interpretation of date parts. For example, if "06-01-07" represents June 1, 2007 then specify "MDY" for the date order.

> **i** Note
>
> If you change the default setting for date_order, you can also change the date_format and timestamp_format options to reflect this change.

Different values determine how the date 10/11/12 is interpreted:

Date order	Interpretation
MDY	October 11 2012
YMD	November 12 2010
DMY	November 10 2012

Use the nearest_century option to control the interpretation of two-digit years in string-to-date conversions.

The following SQL query returns 2010-11-12 using the default date_order and nearest_century settings.

```
SELECT CAST( CAST( '10/11/12' AS DATE ) AS VARCHAR(15) );
```

Related Information

Search Conditions That Compare Dates
Open Client/jConnect TDS Compatibility Options [page 874]
Server Options Changed by ODBC
date_format Option [page 716]
timestamp_format Option [page 850]
sp_tsql_environment System Procedure

1.5.6.7.40 db_publisher Option

Stores the user ID of the database publisher, if any.

Allowed Values

-1, NULL, or any user ID

Default

-1

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	No	No	No

Remarks

Setting this option takes effect immediately.

If the publisher is already set, you must reset this option to -1 or NULL before setting it to another user ID.

Changing the publisher updates the value for the CURRENT PUBLISHER special value.

You can also set and reset the value db_publisher using the GRANT PUBLISH and REVOKE PUBLISH statements, respectively.

Related Information

PUBLISH Privilege
GRANT PUBLISH Statement [SQL Remote]
REVOKE PUBLISH Statement [SQL Remote]
CURRENT PUBLISHER Special Value

1.5.6.7.41 debug_messages Option

Controls whether MESSAGE statements that include a DEBUG ONLY clause are executed.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option allows you to control the behavior of debugging messages in stored procedures and triggers that contain a MESSAGE statement with the DEBUG ONLY clause specified. By default, this option is set to Off and debugging messages do not appear when the MESSAGE statement is executed. By setting debug_messages to On, you can enable the debugging messages in all stored procedures and triggers.

> **i** Note
>
> DEBUG ONLY messages are inexpensive when the debug_messages option is set to Off, so these statements can usually be left in stored procedures on a production system. However, use them sparingly in locations where they would be executed frequently; otherwise, they can result in a small performance penalty.

Related Information

MESSAGE Statement

1.5.6.7.42 dedicated_task Option

Dedicates a request handling task to handling requests from a single connection.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	No	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	Yes (current connection only), with SET ANY SYSTEM OPTION	No

Remarks

When the dedicated_task connection option is set to On, a request handling task is dedicated exclusively to handling requests for the connection. By pre-establishing a connection with this option enabled, you can gather information about the state of the database server if it becomes otherwise unresponsive.

Example

```
DROP USER admin;
CREATE USER ADMIN IDENTIFIED BY sql;
GRANT SET ANY SYSTEM OPTION TO admin;
CREATE OR REPLACE PROCEDURE dba.admin_login_procedure ()
BEGIN
    SET TEMPORARY OPTION dedicated_task='on';
END;
GRANT EXECUTE ON dba.admin_login_procedure TO ADMIN;
SET OPTION admin.login_procedure = 'dba.admin_login_procedure';
```

1.5.6.7.43 default_dbspace Option

Changes the default dbspace in which tables are created.

Allowed Values

String

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

For each database, you can create up to 12 dbspaces in addition to the system (main) dbspace. When a table is created without specifying a dbspace, the dbspace named by this option setting is used. If this option is not set, is set to the empty string, or is set to system, then the system dbspace is used.

When you create temporary tables or indexes, they are always placed in the TEMPORARY dbspace, regardless of the setting of the default_dbspace option. If you specify the IN clause when creating a base table, the dbspace specified by the IN clause is used, rather than the dbspace specified by the default_dbspace option.

If all tables are created in a location other than the system dbspace, then the system dbspace is only used for the checkpoint log and system tables. This is useful to put the checkpoint log on a separate disk from the rest of your database objects for performance reasons. You can place the checkpoint log in a separate disk by changing all CREATE TABLE statements to specify the dbspace, or by changing this option before creating any tables.

Example

In the following example, a new dbspace named MyLibrary is created. The default dbspace is then set to the MyLibrary dbspace and the table LibraryBooks is stored in the MyLibrary dbspace instead of the system dbspace.

```
CREATE DBSPACE MyLibrary
AS 'c:\\dbfiles\\library.db';
SET OPTION default_dbspace = 'MyLibrary';
CREATE TABLE LibraryBooks (
  title CHAR(100),
  author CHAR(50),
  isbn CHAR(30),
);
```

Related Information

Additional Dbspaces Considerations [page 283]
Dbspace Creation [page 285]
CREATE DBSPACE Statement

ALTER DBSPACE Statement
-ds Database Option [page 549]

1.5.6.7.44 default_timestamp_increment Option

Specifies the number of microseconds to add to a column that has a default value of TIMESTAMP or UTC TIMESTAMP to keep values in the column unique when a row containing the column is inserted or updated.

Allowed Values

Integer, between 1 and 1000000 inclusive

Default

1

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

Since a TIMESTAMP value is precise to six decimal places in SQL Anywhere, by default 1 microsecond (0.000001 of a second) is added to differentiate between two identical TIMESTAMP or TIMESTAMP WITH TIME ZONE values.

Settings for this database option do not apply to TIMESTAMP columns that have been declared with the DEFAULT CURRENT TIMESTAMP attribute.

When several rows containing a column with the DEFAULT TIMESTAMP or DEFAULT UTC TIMESTAMP attribute are updated and the column itself is not explicitly updated, the column receives a unique value for each row being updated. For example, three rows containing a column of type TIMESTAMP WITH TIME ZONE with the

DEFAULT UTC TIMESTAMP attribute are updated. The rows are updated at '2011-04-01 12:47:52.724000+00:00', however each row contains a unique UTC TIMESTAMP value.

```
2011-04-01 12:47:52.724000+00:00
2011-04-01 12:47:52.724001+00:00
2011-04-01 12:47:52.724002+00:00
```

Some software, such as Microsoft Access, truncates TIMESTAMP values to three decimal places, making valid comparisons a problem. You can set the truncate_timestamp_values option to On to specify the number of decimal place values the database server stores to maintain compatibility.

For MobiLink synchronization, if you are going to set this option, set it before performing the first synchronization.

Related Information

truncate_timestamp_values Option [page 855]
TIMESTAMP Special Value
UTC TIMESTAMP Special Value

1.5.6.7.45 delayed_commit_timeout Option

Specifies the maximum delay between an application executing a COMMIT and the COMMIT actually being written to disk when the delayed_commits option is set to On.

Allowed Values

Integer, in milliseconds

Default

500

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option has meaning only when delayed_commits is set to On. It governs when a COMMIT entry in the transaction log is written to disk. With delayed_commits set to On, the database server waits for the number of milliseconds set in the delayed_commit_timeout option for other connections to fill a page of the log before writing the current page contents to disk.

Related Information

delayed_commits Option [page 727]
cooperative_commit_timeout Option [Deprecated] [page 712]
cooperative_commits Option [Deprecated] [page 713]
COMMIT Statement

1.5.6.7.46 delayed_commits Option

Determines when the database server returns control to an application following a COMMIT.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

Off corresponds to ISO COMMIT behavior.

When set to On, the database server replies to a COMMIT statement immediately instead of waiting until the transaction log entry for the COMMIT has been written to disk. When set to Off, the application must wait until the COMMIT is written to disk.

When this option is On, the log is written to disk when the log page is full or according to the delayed_commit_timeout option setting, whichever is first. There is a slight chance that a transaction may be lost even though committed if a system failure occurs after the database server replies to a COMMIT, but before the page is written to disk. Setting delayed_commits to On, and the delayed_commit_timeout option to a high value, promotes a quick response time at the slight risk of losing a committed transaction during recovery.

If both cooperative_commits and delayed_commits are set to On, and if the cooperative_commit_timeout interval passes without the pages getting written, the application is resumed (as if the commit had worked), and the remaining interval (delayed_commit_timeout - cooperative_commit_timeout) is used as a delayed_commits interval after which the pages are written, even if they are not full.

Related Information

cooperative_commit_timeout Option [Deprecated] [page 712]
cooperative_commits Option [Deprecated] [page 713]
delayed_commit_timeout Option [page 726]
COMMIT Statement

1.5.6.7.47 delete_old_logs Option [MobiLink][SQL Remote]

Controls whether transaction logs are deleted when their transactions have been replicated or synchronized.

Allowed Values

On, Off, Delay, n days

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

SQL Anywhere MobiLink clients and SQL Remote use this option. The default setting is Off. When it is set to On, each old transaction log is deleted when all the changes it contains have been replicated or synchronized successfully. When it is set to DELAY, transaction logs with a last modified timestamp indicating that they were created on the current day are not deleted, even if all changes have been sent and confirmed. When set to n days, logs that were created before n days ago are deleted. An invalid value is interpreted as Off and no error or warning is reported.

Example

If, on January 18 you run dbmlsync against a remote database that has set the delete_old_logs option to ten days, dbmlsync deletes offline transaction logs that were created on or before January 7. The remote database would set the option as follows:

```
SET OPTION delete_old_logs = '10 days';
```

Related Information

The Transaction Log [page 292]
SQL Anywhere Client Logs
Maintaining Transaction Logs for Remote Databases (Command Line)
BACKUP DATABASE Statement
Transaction Log Utility (dblog) [page 1231]
MirrorLogDirectory (mld) Extended Option
SQL Remote Options

1.5.6.7.48 disk_sandbox Option

Controls whether the read-write file operations of the database are restricted to the directory where the main database file is located and any subdirectories of this directory.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SECURITY OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SECURITY OPTION	No	No

Remarks

Setting this option to On restricts the read-write file operations on the database to the directory where the main database file is located and any subdirectories of this directory. The setting persists across sessions.

Related Information

Disk Sandboxing [page 1687]
-sbx Database Server Option [page 491]
-sbx Database Option [page 559]
START DATABASE Statement
sa_db_option System Procedure

1.5.6.7.49 divide_by_zero_error Option

Controls the reporting of division by zero.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option indicates whether division by zero is reported as an error. If the option is set On, then division by zero results in an error with SQLSTATE '22012'.

If the option is set Off, division by zero is not an error. Instead, a NULL is returned.

Related Information

Materialized Views Restrictions

1.5.6.7.50 escape_character Option (Reserved for System Use)

Reserved for system use.

Default

On

A temporary setting for the current user is established by SAP Open Client and jConnect TDS connections ('Off').

Remarks

Do not change the setting of this option.

Related Information

Open Client/jConnect TDS Compatibility Options [page 874]

1.5.6.7.51 exclude_operators Option (Reserved for System Use)

Reserved for system use. Do not change the setting of this option.

1.5.6.7.52 extended_join_syntax Option

Controls whether queries with duplicate correlation names syntax for multi-table joins are allowed, or reported as an error.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

If this option is set to On, then the database server allows duplicate correlation names to be used in the null-supplying side of outer joins. All tables or views specified with the same correlation name are interpreted as the same instance of the table or view.

The following FROM clause illustrates the SQL Anywhere interpretation of a join using duplicate correlation names where C1 and C2 are search conditions:

```
( R left outer join T on ( C1 ), T  join S on ( C2 ) )
```

If the option is set to On, this join is interpreted as follows:

```
( R left outer join T on ( C1 ) ) join S on ( C2 )
```

If the option is set to Off, SQLCODE -137 is generated.

> **i** Note
>
> To see the result of eliminating duplicate correlation names, you can view the rewritten statement using the REWRITE function with the second argument set to ANSI.

Related Information

Duplicate Correlation Names in Joins (Star Joins)
REWRITE Function [Miscellaneous]

1.5.6.7.53 extern_login_credentials Option

Controls whether the external login credentials of the session user or the executing (effective) user are used when making remote connections. This option is provided for backwards compatibility. For security reasons, do not specify this option.

Allowed Values

Login_user, Effective_user

Default

Effective_user

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SECURITY OPTION	No	No

	PUBLIC role	For current user	For other users
Allowed to set temporarily?	Yes, with SET ANY SECURITY OPTION	No	No

Remarks

This option specifies whether remote data access connections are performed using the external login credentials of the session user or the current executing user. The session user is the user who is connected; whereas the current executing user can be different based on stored procedure calls.

Login_user

When Login_user is specified, the database server always uses the session user's external login credentials when making remote data access connections, regardless of what the current executing user is.

Effective_user

When Effective_user is specified, the database server respects the setting of the current executing user. Remote connections are made based on the external login credentials of the current executing user. Specifying Effective_user can result in multiple (one per current executing user) remote connections for the same local connection.

Related Information

Procedures and Functions Running with Owner or Invoker Privileges
Remote Servers and Remote Table Mappings
CREATE PROCEDURE Statement
CREATE SERVER Statement

1.5.6.7.54 external_remote_options Option [SQL Remote]

Indicates where the message link parameters should be stored.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

SQL Remote uses this option to indicate whether the message link parameters are stored in the database (Off) or externally (On).

Related Information

Remote Message Type Control Parameters

1.5.6.7.55 fire_triggers Option

Controls whether triggers are fired in the database.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	Yes, with SET ANY SYSTEM OPTION	Yes, with SET ANY SYSTEM OPTION
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	Yes (current connection only), with SET ANY SYSTEM OPTION	No

Remarks

When set to On, triggers are fired. When set to Off, no triggers are fired, including referential integrity triggers (such as cascading updates and deletes). The -gf option overrides this option, which turns off all trigger firing regardless of the fire_triggers setting.

This option is relevant when replicating data from Adaptive Server Enterprise to SQL Anywhere because all actions from Adaptive Server Enterprise transaction logs are replicated to SQL Anywhere, including actions performed by triggers.

Related Information

Triggers
Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
-gf Database Server Option [page 432]

1.5.6.7.56 first_day_of_week Option

Sets the numbering of the days of the week.

Allowed Values

1, 2, 3, 4, 5, 6, 7

Default

7

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The values have the following meaning:

Value	Meaning
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday
7	Sunday

The default is 7, which means Sunday is the first day of the week.

The value specified by this option affects the result of the DATEPART function when obtaining a weekday value. You can also change the first day of week using the DATEFIRST option in the T-SQL SET statement.

The value specified by this option does not affect the result of the DOW function. For example, even if the first day of the week is set to Monday, the DOW function returns a 2 for Monday.

Related Information

DATEPART Function [Date and Time]
DOW Function [Date and Time]
SET Statement [T-SQL]

1.5.6.7.57 for_xml_null_treatment Option

Controls the treatment of NULL values in queries that use the FOR XML clause.

Allowed Values

Empty, Omit

Default

Omit

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

If you execute a query that includes the FOR XML clause, the for_xml_null_treatment option determines how NULL values are treated. By default, elements and attributes that contain NULL values are omitted from the result. Setting this option to Empty generates empty elements or attributes if the value is NULL.

Related Information

Use of the FOR XML Clause to Retrieve Query Results as XML
SELECT Statement

1.5.6.7.58 force_view_creation Option (Reserved for System Use)

Reserved for system use. Do not change the setting of this option.

> ⚠ Caution
>
> The force_view_creation option should only be used within a `reload.sql` script. This option is used by the Unload utility (dbunload) and should not be set explicitly.

1.5.6.7.59 global_database_id Option

Controls the range of values for columns created with DEFAULT GLOBAL AUTOINCREMENT.

For use in generating unique primary keys in a replication environment.

Allowed Values

Non-negative integer

Default

2147483647

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

The value you specify for this option is the starting value. For columns created with DEFAULT GLOBAL AUTOINCREMENT, when a row is inserted into the table that does not include a value for the DEFAULT GLOBAL

AUTOINCREMENT column, the database server generates a value for the column. The value is determined by the global_database_id value and the partition size for the column.

Setting global_database_id to the default value indicates that DEFAULT GLOBAL AUTOINCREMENT is disabled. In this case NULL is generated as a default.

You can find the value of the option in the current database using the following statement:

```
SELECT DB_PROPERTY( 'GlobalDBID' );
```

This feature is of particular use in replication environments to ensure unique primary keys.

Example

The following example sets the database identification number to 100.

```
SET OPTION PUBLIC.global_database_id = '100';
```

Related Information

The GLOBAL AUTOINCREMENT Default
Reloading Tables with AUTOINCREMENT Columns
Global Database IDs
CREATE TABLE Statement
Duplicate Primary Key Errors

1.5.6.7.60 http_connection_pool_basesize Option

Specifies the nominal threshold size of database connections.

Allowed Values

Integer, between 0 and 1000, inclusive

Default

10

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

Changing this option takes effect within 5 seconds during the next purge cycle. Any unused database connections having exceeded http_connection_pool_timeout within the http_connection_pool_basesize threshold are deleted. When the size of a connection pool exceeds the http_connection_pool_basesize threshold, connections are deleted at a more aggressive rate.

Specifying a value of 0 purges the HTTP connection pools within 5 seconds of setting the option and disables HTTP connection pooling.

Unused excess connections are removed within half the timeout interval, half of the remainder are purged if unused within a quarter timeout interval, and so on. The following database properties have been added to determine the operating efficiency of the connection pools within a database:

- HttpConnPoolCachedCount
- HttpConnPoolHits
- HttpConnPoolMisses
- HttpConnPoolSteals

Related Information

HTTP Web Services
http_connection_pool_timeout Option [page 742]
List of Database Server Properties [page 900]

1.5.6.7.61 http_connection_pool_timeout Option

Specifies the maximum duration that an unused connection can be retained in the connection pool.

Allowed Values

Integer, between 1 and 86400, inclusive

Default

60

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

Excess connections are deleted at an increasing rate based on this value if the pool size exceeds http_connection_pool_basesize.

Changing this option takes effect within 5 seconds during the next purge cycle. Any unused database connections having exceeded http_connection_pool_timeout within the http_connection_pool_basesize threshold are deleted. When the size of a connection pool exceeds the http_connection_pool_basesize threshold, connections are deleted at a more aggressive rate.

Unused excess connections are removed within half the timeout interval, half of the remainder are purged if unused within a quarter timeout interval, and so on. The following database properties have been added to determine the operating efficiency of the connection pools within a database:

- HttpConnPoolCachedCount
- HttpConnPoolHits
- HttpConnPoolMisses
- HttpConnPoolSteals

Related Information

HTTP Web Services
http_connection_pool_basesize Option [page 741]
List of Database Server Properties [page 900]

1.5.6.7.62 http_session_timeout Option

Specifies the default timeout duration, in minutes, that the HTTP session persists during inactivity.

Allowed Values

Integer (1 to 525600)

Default

30

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

Setting this database option to another value sets a new default session timeout that is applied to all later HTTP sessions. An HTTP session can override this default by calling the sa_set_http_option system procedure to set the SessionTimeout within a particular HTTP session.

Related Information

HTTP Session Management on an HTTP Server
HTTP Web Services
sa_set_http_option System Procedure

1.5.6.7.63 integrated_server_name Option

Specifies the name of the Domain Controller server used for looking up Windows user group membership for integrated user authentication (logins).

Allowed Values

String

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SECURITY OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SECURITY OPTION	No	No

Remarks

This option allows a user with SET ANY SECURITY OPTION system privilege to specify the name of the Domain Controller server that is used to look up group membership when using Windows user groups for Integrated logins. By default, the computer that the database server is running on is used for verifying group membership.

Example

The following example specifies that group membership is verified on the computer server-1.

```
SET OPTION PUBLIC.integrated_server_name = '\\\\server-1';
```

Related Information

Integrated Logins for Microsoft Windows User Groups [page 136]

1.5.6.7.64 isolation_level Option

Controls the locking isolation level.

Allowed Values

0, 1, 2, 3, Snapshot, Statement-snapshot, Readonly-statement-snapshot

Default

0

A temporary setting for the current user is established by the JDBC driver (0).

A temporary setting for the current user is established by SAP Open Client and jConnect TDS connections (1).

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option controls the locking isolation level as follows:

0

Allow dirty reads, non-repeatable reads, and phantom rows.

1

Prevent dirty reads. Allow non-repeatable reads and phantom rows.

2

Prevent dirty reads and non-repeatable reads. Allow phantom rows.

3

Serializable. Prevent dirty reads, non-repeatable reads, and phantom rows.

Snapshot

Use a snapshot of committed data from the time when the first row is read or updated by the transaction.

Statement-snapshot

For each statement, use a snapshot of committed data from the time when the first row is read from the database. Non-repeatable reads and phantom rows can occur within a transaction, but not within a single statement.

Readonly-statement-snapshot

For read-only statements, use a snapshot of committed data from the time when the first row is read from the database. Non-repeatable reads and phantom rows can occur within a transaction, but not within a single statement. For updatable statements, use the isolation level specified by the updatable_statement_isolation option (can be one of 0 (the default), 1, 2, or 3).

Set the allow_snapshot_isolation option to On to use the Snapshot, Statement-snapshot, or Readonly-statement-snapshot settings.

Queries running at isolation level Snapshot, Statement-snapshot, or Readonly-statement-snapshot see a snapshot of a committed state of the database.

You can override any temporary or PUBLIC settings for this option within individual INSERT, UPDATE, DELETE, SELECT, UNION, EXCEPT, and INTERSECT statements by including an OPTION clause in the statement.

Related Information

Isolation Levels and Consistency
Snapshot Isolation
Guidelines for Choosing Isolation Levels
Open Client/jConnect TDS Compatibility Options [page 874]
Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
sp_tsql_environment System Procedure
allow_snapshot_isolation Option [page 675]
updatable_statement_isolation Option [page 860]
BEGIN TRANSACTION Statement [T-SQL]

1.5.6.7.65 java_class_path Option

Specifies an additional set of directories or JAR files in which to search for classes.

Allowed Values

String

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

All Java classes and JAR files used with Java in the database should be installed in the database. If the database is moved to a different computer or operating system, the Java classes and JAR files move with it. Another benefit of installing classes and JAR files into the database is that the database server class loader can fetch the classes and resources from the database, allowing each connection that is using Java in the database to have its own instance of these classes and its own copy of static variables within these classes.

However, when a class or JAR file must be loaded by the system class loader, it can be specified with the java_class_path database option or the -cp dbeng17/dbsrv17 server option. Both options add classes and JAR files to the classpath that the database server builds for launching the Java VM. The java_class_path database option is useful when the server is running multiple databases and each database has a different set of JARs and directories that must be loaded by the system class loader. The -cp database server option is useful when all databases on the server require the same classes or JAR files.

Related Information

Java in the Database
-cp Database Server Option [page 412]

1.5.6.7.66 java_location Option

Specifies the path of the Java VM for the database.

Allowed Values

String

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

By default, this option contains an empty string. In this case, the database server searches the JAVA_HOME environment variable, the path, and other locations for the Java VM. The JavaVM database property allows you to query which Java VM the database server uses if the java_location option is not set.

Related Information

Lesson 2: Selecting a Java VM

java_vm_options Option [page 750]

1.5.6.7.67 java_main_userid Option (Deprecated)

This option is deprecated.

1.5.6.7.68 java_vm_options Option

Specifies command-line options that the database server uses when it launches the Java VM.

Allowed Values

String

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

This option lets you specify options that the database server uses when launching the Java VM specified by the java_location option. These additional options can be used to set up the Java VM for debugging purposes or to run as a service on Unix platforms. Sometimes additional options are required to use the Java VM in 64-bit mode instead of 32-bit mode.

Example

The following example uses the java_vm_options option to keep the Java VM running on Unix when the database server is started as a service and the user must log out:

```
SET OPTION PUBLIC.java_vm_options = '-Xrs';
```

The following example instructs the Java VM to use 64-bit mode on HP-UX:

```
SET OPTION PUBLIC.java_vm_options = '-d64';
```

Related Information

Lesson 2: Selecting a Java VM
java_location Option [page 749]

1.5.6.7.69 log_deadlocks Option

Controls whether deadlock reporting is turned on or off.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

When this option is set to On, the database server logs information about deadlocks in an internal buffer. The size of the buffer is fixed at 10000 bytes. You can view the deadlock information using the sa_report_deadlocks stored procedure. The contents of the buffer are retained when this option is set to Off.

When deadlock occurs, information is reported for only those connections involved in the deadlock. The order in which connections are reported is based on which connection is waiting for which row. For thread deadlocks, information is reported about all connections.

When you have deadlock reporting turned on, you can also use the Deadlock system event to take action when a deadlock occurs.

You can also change the setting of this option by using the DeadlockLogging property with the sa_server_option system procedure.

Related Information

System Events [page 1001]
Example: Determining Who Is Blocked in a Deadlock (SQL)
Tutorial: Diagnosing Blocked Connections and Deadlocks (Profiler) [page 1496]
sa_report_deadlocks System Procedure
sa_server_option System Procedure

1.5.6.7.70 login_mode Option

Controls the use of Standard, Integrated, Kerberos, LDAP, and PAM user authentication for the database.

Allowed Values

One or more of: Standard, Integrated, Kerberos, LDAPUA, PAMUA, CloudAdmin, Mixed (deprecated)

Default

Standard

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SECURITY OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SECURITY OPTION	No	No

Remarks

This option specifies whether Standard, Integrated, Kerberos, LDAP, PAM, and CloudAdmin user authentication is permitted. One or more of the following login modes are accepted (the values are case insensitive):

> **i** Note
>
> Do not set Integrated, Kerberos, LDAPUA, or PAMUA as a permanent login_mode as this setting can allow a user unauthorized access to the database if they obtain a copy of the database.

Standard

Standard user authentication is permitted. This value is the default setting. Connections that use standard user authentication include both a user ID and password, and do not use the Integrated or Kerberos connection parameters.

Integrated

Integrated user authentication is permitted.

Kerberos

Kerberos user authentication is permitted.

LDAPUA

LDAP (Lightweight Directory Access Protocol) user authentication is permitted. Connections that use LDAP user authentication include both a user ID and password, and do not use the Integrated or Kerberos connection parameters.

If a user's hashed password has changed, then it is updated in the SYSUSER table of the database when the user logs in to the database using LDAPUA if the ldap_failover_to_std option is set to ON.

When using LDAPUA, the password control rules of the login policy are ignored.

PAMUA PAM (Pluggable Authentication Modules) user authentication is permitted. If a user's hashed password has changed, then it is updated in the SYSUSER table of the database when the user logs into the database using PAMUA when the pam_failover_to_std option is set to ON.

When using PAMUA, the password control rules of the login policy are ignored.

CloudAdmin

This login mode is for internal use in the cloud.

Mixed (deprecated)

This value is equivalent to specifying Standard,Integrated.

If you specify multiple login modes, then the database server allows all the specified modes.

> ⚠ Caution
>
> Setting the login_mode database option to not allow Standard user authentication restricts connections to only those users who have been granted an Integrated, Kerberos, LDAP, or PAM login mapping. Attempting to connect with a standard database user ID and password generates an error. The only exception to this rule are users with MANAGE ANY USER privilege.

Specify multiple values in a comma-separated list. This list cannot contain white space. For example, the following setting allows both Standard and Integrated logins:

```
SET OPTION PUBLIC.login_mode = 'Standard,Integrated';
```

> ⚠ Caution
>
> If a database file is not secured and can be copied by unauthorized users, then the temporary PUBLIC login_mode option should be used (for Integrated, Kerberos, LDAP, and PAM user authentication). This way, Integrated, Kerberos, LDAPUA, and PAMUA logins are not supported by default if the file is copied.

Example

Enable only Integrated user authentication (Standard, Kerberos, LDAP, and PAM user authentication fail):

```
SET OPTION PUBLIC.login_mode = 'Integrated';
```

Enable Standard and Kerberos user authentication (Integrated, LDAP, and PAM user authentication fail):

```
SET OPTION PUBLIC.login_mode = 'Standard,Kerberos';
```

Enable Standard, Integrated, Kerberos, LDAP, and PAM user authentication:

```
SET OPTION PUBLIC.login_mode = 'Standard,Integrated,Kerberos,LDAPUA,PAMUA';
```

Related Information

Microsoft Windows Integrated Login [page 129]
Kerberos User Authentication [page 159]
Security: User IDs [page 1684]
Security: Use Login Modes to Secure the Database [page 171]
-al Database Server Option [page 399]

1.5.6.7.71 login_procedure Option

Specifies a stored procedure that is called when a user connects via a database login or web service.

Allowed Values

String

Default

sp_login_environment

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SECURITY OPTION	Yes, with SET ANY SECURITY OPTION	Yes, with SET ANY SECURITY OPTION
Allowed to set temporarily?	Yes, with SET ANY SECURITY OPTION	Yes (current connection only), with SET ANY SECURITY OPTION	No

Remarks

By default, the sp_login_environment stored procedure is called. The login procedure is called after all the checks have been performed to verify that the connection is valid. The procedure specified by the login_procedure option is not executed for event connections, but it is executed for normal login and web service connections.

You can create a custom login procedure by creating a new stored procedure and setting login_procedure to the owner and name of this new procedure.

This custom procedure should call the dbo.sp_login_environment stored procedure at some point during its execution since it detects when a TDS connection occurs and sets appropriate database options using the dbo.sp_tsql_environment stored procedure. Failure to do so can break TDS-based connections. It is not a good practice to revise either of the dbo.sp_login_environment or dbo.sp_tsql_environment stored procedures.

A password expired error message with SQLSTATE '08WA0' can be signaled by a user-defined login procedure to indicate to a user that their password has expired. Signaling the error allows applications to check for the error and process expired passwords. However, it is better to use a login policy to implement password expiry and not a login procedure that returns the expired password error message.

If you use the NewPassword=* connection parameter, signaling this error is required for the client libraries to prompt for a new password. If the procedure signals SQLSTATE '28000' (invalid user ID or password) or SQLSTATE '08WA0' (expired password), or the procedure raises an error with RAISERROR, the login fails and an error is returned to the user. If you signal any other error or if another error occurs, then the user login is successful and a message is written to the database server message log.

Example

The following example shows how you can limit the total number of database connections.

```
CREATE PROCEDURE DBA.login_check( )
   BEGIN
      DECLARE INVALID_LOGON EXCEPTION FOR SQLSTATE '28000';
      // Allow a maximum of 3 concurrent connections
      IF( DB_PROPERTY( 'ConnCount' ) > 3 ) THEN
          SIGNAL INVALID_LOGON;
      ELSE
          CALL dbo.sp_login_environment;
      END IF;
   END
go
GRANT EXECUTE ON DBA.login_check TO PUBLIC
go
SET OPTION PUBLIC.login_procedure='DBA.login_check'
go
```

The following example shows how you can block connection attempts if the number of failed connections for a user exceeds 3 within a 30 minute period. All blocked attempts during the block out period receive an invalid password error and are logged as failures. The log is kept long enough for a DBA to analyze it.

```
CREATE TABLE DBA.ConnectionFailure(
    pk INT PRIMARY KEY DEFAULT AUTOINCREMENT,
    user_name CHAR(128) NOT NULL,
    tm TIMESTAMP NOT NULL DEFAULT CURRENT TIMESTAMP
)
go
CREATE INDEX ConnFailTime ON DBA.ConnectionFailure(
    user_name, tm )
go
CREATE EVENT ConnFail TYPE ConnectFailed
HANDLER
BEGIN
    DECLARE usr CHAR(128);
    SET usr = event_parameter( 'User' );
    // Put a limit on the number of failures logged.
    IF (SELECT COUNT(*) FROM DBA.ConnectionFailure
        WHERE user_name = usr
        AND tm >= DATEADD( minute, -30, CURRENT TIMESTAMP )
        ) < 20
    THEN
        INSERT INTO DBA.ConnectionFailure( user_name )
            VALUES( usr );
        COMMIT;
        // Delete failures older than 7 days.
        DELETE DBA.ConnectionFailure
        WHERE user_name = usr
        AND tm < DATEADD( day, -7, CURRENT TIMESTAMP );
        COMMIT;
    END IF;
END
```

```
go
CREATE PROCEDURE DBA.login_check( )
BEGIN
    DECLARE usr CHAR(128);
    DECLARE INVALID_LOGON EXCEPTION FOR SQLSTATE '28000';
    SET usr = CONNECTION_PROPERTY( 'Userid' );
    // Block connection attempts from this user
    // if 3 or more failed connection attempts have occurred
    // within the past 30 minutes.
    IF ( SELECT COUNT( * ) FROM DBA.ConnectionFailure
        WHERE user_name = usr
        AND tm >= DATEADD( minute, -30,
            CURRENT TIMESTAMP ) ) >= 3 THEN
        SIGNAL INVALID_LOGON;
    ELSE
        CALL dbo.sp_login_environment;
    END IF;
END
go
GRANT EXECUTE ON DBA.login_check TO PUBLIC
go
SET OPTION PUBLIC.login_procedure='DBA.login_check'
go
```

The following example shows how to signal an error indicating that the user's password has expired. However, it is better to use a login policy to implement password expiry notification.

```
CREATE PROCEDURE DBA.check_expired_login( )
BEGIN
  DECLARE PASSWORD_EXPIRED EXCEPTION FOR SQLSTATE '08WA0';
  IF( condition-to-check-for-expired-password ) THEN
      SIGNAL PASSWORD_EXPIRED;
  ELSE
      CALL dbo.sp_login_environment;
  END IF;
END;
```

Related Information

Security: Passwords [page 1685]
Login Policies [page 640]
post_login_procedure Option [page 791]
sp_login_environment System Procedure
sp_tsql_environment System Procedure
NewPassword (NEWPWD) Connection Parameter [page 98]
CREATE PROCEDURE Statement
RAISERROR Statement

1.5.6.7.72 materialized_view_optimization Option

Controls how materialized views are used by the optimizer to answer queries efficiently.

Allowed Values

Disabled, *Fresh*, *Stale*, N { *Minute*[*s*] | *Hour*[*s*] | *Day*[*s*] | *Week*[*s*] | *Month*[*s*] }

Default

Stale

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The materialized_view_optimization option lets you specify the circumstances under which the optimizer can use stale materialized views.

Data in a materialized view becomes stale when data in any of the base tables referenced by the materialized view is updated. Consider the acceptable degree of data staleness when deciding the refresh frequency for the materialized view, and the time it takes to refresh the view, since the view is not available for querying during the refresh process. Also consider whether it is acceptable for the database server to return results that may not reflect the current state of the database. You can choose from the following settings for this option:

Disabled

Do not use materialized views for query optimization.

Fresh

Use a materialized view only if it is fresh (data in underlying tables has not been modified since the view was last refreshed).

Stale

Use materialized views even if they are stale. This value is the default setting.

N { Minute[s] | Hour[s] | Day[s] | Week[s] | Month[s] }

Use fresh and stale materialized views, as long as the stale materialized views have been refreshed within the specified time period. Values specified in minutes must be less than 2^{31} minutes. The database server treats a week as 7 days and a month as 30 days.

When a query directly references a materialized view, the view is used regardless of staleness; the materialized_view_optimization option has no effect in this case.

Related Information

Materialized Views
Advanced: Settings Controlling Data Staleness for Materialized Views

1.5.6.7.73 max_client_statements_cached Option

Controls the number of statements cached by the client.

Allowed Values

Integer, 0 to 100

Default

10

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

Client statement caching reduces database requests and statement prepares when identical SQL statements are prepared multiple times. When the same SQL text is prepared and dropped repeatedly, the client caches the statement, leaving it prepared on the database server, even after it has been dropped by the application. Caching the statement saves the database server the extra work of dropping and re-preparing the statement. If a schema change occurs, a database option setting changes, or a DROP VARIABLE statement is executed, the prepared statement is dropped automatically and is prepared again the next time the SQL statement is executed, ensuring that a cached statement that could cause incorrect behavior is never reused.

This option specifies the maximum number of statements that can remain prepared (cached). Cached statements are not counted toward the max_statement_count resource governor.

The setting of this option applies to connections made using Embedded SQL, ODBC, OLE DB, ADO.NET, and the SQL Anywhere JDBC driver. It does not apply to SAP Open Client, jConnect, or HTTP connections.

Setting this option to 0 disables client statement caching. Increasing this value has the potential to improve performance if the application is repeatedly preparing and dropping more than ten of the same SQL statements. For example, if an application loops through 25 SQL statements, preparing and dropping them each iteration through the loop, and in each iteration each of these SQL statements has the exact same text, setting this option to 25 may improve performance.

Increasing the value of this option increases memory use on the client and places more cache pressure on the database server. If a significant number of cached statements cannot be reused because of schema changes or option settings, statement caching is disabled automatically for the connection. If statement caching is automatically turned off, the client periodically turns statement caching on again to re-evaluate the decision and determine whether re-enabling statement caching would be beneficial.

Client statement caching could deliver unexpected results in the following situation:

1. A statement is prepared and described and its describe returns that the statement has no result.

   ```
   CREATE OR REPLACE FUNCTION test() RETURNS INT BEGIN RETURN 1;
   END;
   CALL test();
   ```

2. A DDL statement causes the same statement text (in this example, the CALL statement) to now return a result set on the same connection. For example:

   ```
   CREATE OR REPLACE PROCEDURE test() BEGIN SELECT 2;
   END;
   CALL test();
   ```

   When client statement caching is enabled, the second `CALL test()` statement is described incorrectly.

If the log is being analyzed using the `tracetime.pl` Perl script, the max_client_statements_cached option should be set to 0 to disable client statement caching while the request log is captured.

Related Information

max_statement_count Option [page 769]

1.5.6.7.74 max_connections Option

Controls the number of concurrent connections that are allowed to the database.

Allowed Values

Integer (greater than or equal to 0), NULL

Default

The default limit of concurrent connections for the network database server is equal to the maximum number of connections allowed by your per-seat license or 32766 connections if you have a per-core license.

The default limit of concurrent connections for the personal server is 10, of which a minimum of 3 must be standard connections.

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

The max_connections option specifies the number of concurrent connections that are allowed to the database. Once the limit is reached:

- Subsequent standard connection attempts fail until the number of connections to the database falls below the specified limit.
- Subsequent HTTP/HTTPS connection attempts are queued until the number of connections falls below the specified limit, or the queued connections time out.
  While there are connections in the HTTP/HTTPS queue, there is no opportunity for a standard connection to replace an HTTP/HTTPS connection. A user wanting to make a standard connection could wait indefinitely unless some of the database connections are reserved for standard connections. Reserve connections by using the reserved_connections option.

Even if the connection limit has been reached, a DBA user with the DROP CONNECTION or SERVER OPERATOR system privilege can make one standard connection to the database to drop connections as needed.

Example

- The following example sets the maximum number of concurrent connections to 1000 for a database running on a network database server.

  ```
  SET OPTION PUBLIC.max_connections=1000;
  ```

- The following example sets the maximum number of concurrent connections to 3 for a database running on a personal database server.

  ```
  SET OPTION PUBLIC.max_connections=3;
  ```

Related Information

Limiting Database Connections [page 251]
SET OPTION Statement

1.5.6.7.75 max_cursor_count Option

Controls a resource governor that limits the maximum number of cursors that a connection can use at once.

Allowed Values

Integer

Default

50

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	Yes, with SET ANY SYSTEM OPTION	Yes, with SET ANY SYSTEM OPTION
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	Yes (current connection only), with SET ANY SYSTEM OPTION	No

Remarks

This resource governor allows a DBA to limit the number of cursors per connection that a user can use. If an operation would exceed the limit for a connection, an error is generated, indicating that the governor for the resource has been exceeded.

If a connection executes a stored procedure, that procedure is executed under the privileges of the procedure owner. However, the resources used by the procedure are assigned to the current connection.

You can remove resource limits by setting the option to 0 (zero).

1.5.6.7.76 max_parallel_statements Option

Specifies the maximum number of statements listed inside the BEGIN PARALLEL WORK statement that the database server can execute in parallel at any time.

Allowed Values

Integer

Default

0 (The database server executes as many statements in parallel as it chooses).

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Setting the max_parallel_statements option to 1 disables the parallel execution of the statements inside a BEGIN PARALLEL WORK statement.

The max_parallel_statements option respects the settings of the -gtc and -gta database server options and the ProcessorAffinity option of the sa_server_option system procedure.

Example

The following example sets the max_parallel_statements option to four.

```
SET TEMPORARY OPTION max_parallel_statements = 4;
```

Related Information

Improving Performance by Executing a List of CREATE INDEX or a List of LOAD TABLE Statements Concurrently [page 1481]
BEGIN PARALLEL WORK Statement
-gtc Database Server Option [page 449]
-gtc Database Server Option [page 449]
-gta Database Server Option [page 447]

1.5.6.7.77 max_plans_cached Option

Specifies the maximum number of execution plans to be stored in a cache.

Allowed Values

Integer

Default

20

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option specifies the maximum number of plans cached for each connection. The optimizer caches the execution plan for queries, INSERT, UPDATE, and DELETE statements that are performed inside stored procedures, functions, and triggers. After a statement in a stored procedure, stored function, or trigger is executed several times by a connection, the optimizer builds a reusable plan for the statement.

Reusable plans do not use the values of host variables for selectivity estimation or rewrite optimizations. As a result of this behavior, the reusable plan can have a higher cost than if the statement was re-optimized. When the cost of the reusable plan is close to the best observed cost for a statement, the optimizer adds the plan to the plan cache.

The cache is cleared when you execute statements, such as CREATE TABLE and DROP TABLE, that modify the table schema. Statements that reference declared temporary tables are not cached.

Setting this option to 0 disables plan caching.

Related Information

Plan Caching

1.5.6.7.78 max_priority Option

Controls the maximum priority level for connections.

Allowed Values

Critical, High, Above Normal, Normal, Below normal, Low, Background

Default

normal

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	Yes, with SET ANY SYSTEM OPTION	Yes, with SET ANY SYSTEM OPTION
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	Yes (current connection only), with SET ANY SYSTEM OPTION	No

Remarks

The scheduling of different priority levels allows all requests to get some CPU time, regardless of the priority level of the request. Higher priority requests get more time slices than lower priority ones.

Related Information

background_priority Option [Deprecated] [page 692]
priority Option [page 798]

1.5.6.7.79 max_query_tasks Option

Specifies the maximum number of server tasks that the database server can use to process a query in parallel.

Allowed Values

Integer

Default

0

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The max_query_tasks option sets the maximum level of parallelism that can be used for any query processing statement. The option sets the number of database server tasks that can be used to process a query in parallel. The default value is 0, which allows the database server to use as many tasks as it chooses. Any other value for the max_query_tasks option sets the maximum number of tasks allowed per query. Setting the max_query_tasks option to 1 disables intra-query parallelism.

The number of tasks the database server can use for all requests is limited by the threshold set using the -gn option at startup. This number is a global maximum for all databases and connections serviced by that server. The number of tasks used for a request is also limited by the number of logical processors available to the database server. For example, setting the processor concurrency to 1 with the -gtc option disables intra-query parallelism.

When enabled, intra-query parallelism is used to process SELECT statements that meet certain qualifications. The presence of an exchange operator in the access plan for a query indicates that intra-query parallelism was used.

You can override any temporary or PUBLIC settings for this option within individual INSERT, UPDATE, DELETE, SELECT, UNION, EXCEPT, and INTERSECT statements by including an OPTION clause in the statement.

Related Information

Threading [page 344]
Database Server Configuration of the Multiprogramming Level [page 348]
-gn Database Server Option [page 437]
-gt Database Server Option [page 446]
-gtc Database Server Option [page 449]

1.5.6.7.80 max_recursive_iterations Option

Limits the maximum number of iterations a recursive common table expression can make.

Allowed Values

Integer

Default

100

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

Computation of a recursive common table expression aborts and an error is generated if the computation fails to complete within the specified number of iterations. Recursive subqueries often geometrically increase the amount of resources required for each additional iteration. Set this option to limit the amount of time and resources that are consumed before infinite recursion is detected, yet permit your recursive common table expressions to work as intended.

Setting this option to 0 disables recursive common table expressions.

Related Information

Recursive Common Table Expressions

1.5.6.7.81 max_statement_count Option

Controls a resource governor that limits the maximum number of prepared statements that a connection can use simultaneously.

Allowed Values

Integer

Default

50

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	Yes, with SET ANY SYSTEM OPTION	Yes, with SET ANY SYSTEM OPTION
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	Yes (current connection only), with SET ANY SYSTEM OPTION	No

Remarks

Applications that use prepared statements can receive the error "Resource governor for 'prepared statements' exceeded" if the prepared statements are not explicitly dropped once they are no longer required. The max_statement_count database option is a resource governor that allows a DBA to limit the number of prepared statements used per connection. If an operation would exceed the limit for a connection, an error is generated, indicating that the governor for the resource has been exceeded.

If a connection executes a stored procedure, that procedure is executed under the privileges of the procedure owner. However, the resources used by the procedure are assigned to the current connection.

The database server maintains data structures for each prepared statement a connection creates. These structures are only freed when the application signals to the database server that the prepared statements are no longer needed or if the connection disconnects. To reduce the statement count for a connection, you must execute the equivalent of a DROP STATEMENT request. The following table lists the statements you can execute for the APIs supported by SQL Anywhere:

Interface	Statement
ADO	RecordSet.Close
ADO.NET	SADataReader.Close or SADataReader.Dispose
Embedded SQL	DROP STATEMENT
Java	resultSet.Close, Statement.Close
ODBC	SQLFreeStmt(hstmt, SQL_DROP) or SQLFreeHandle(SQL_HANDLE_STMT, hstmt)

> **i** Note
>
> In Java and .NET, statements should be dropped explicitly. Do not rely on garbage collection to perform this cleanup because the language routines do not issue server calls to deallocate the statement resources. In addition, there is no guarantee regarding when the garbage collection routines are executed.

If a server needs to support more than the default number of prepared statements at any one time for any one connection, then the max_statement_count setting should be set to a higher value. Note, however, that larger numbers of active prepared statements consume additional server memory. You can disable the prepared statement resource governor entirely by setting the max_statement_count option to 0 (zero), but this is not recommended. Doing so makes the database server vulnerable to shutting down with an out-of-memory condition for applications that do not properly free prepared statements.

Related Information

Prepared Statements
DROP STATEMENT Statement [ESQL]

1.5.6.7.82 max_temp_space Option

Controls the maximum amount of temporary file space a connection can use.

Allowed Values

```
Integer [ k | m | g | p ]
```

Default

0

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	Yes, with SET ANY SYSTEM OPTION	Yes, with SET ANY SYSTEM OPTION
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	Yes (current connection only), with SET ANY SYSTEM OPTION	No

Remarks

This option allows you to specify the maximum amount of temporary file space a connection can use before the request fails because it exceeds the temporary file space limit. The temp_space_limit_check option must be set to On (the default) for the max_temp_space option to take effect.

The default value 0 indicates that there is no fixed limit on the amount of temporary file space a connection can request. Any other value specifies the number of bytes of temporary file space a connection can use. You can use k, m, or g to specify units of kilobytes, megabytes, or gigabytes, respectively. If you use p, the argument is a percentage of the total amount of temporary file space available.

For connections that request temporary file space, the database server checks the limit against the setting of the max_temp_space option to make sure that the request is under the maximum size. If the connection requests more temporary space than is allowed, the request fails and the error SQLSTATE_TEMP_SPACE_LIMIT is generated.

Example

Set a 1-GB limit for a connection:

```
SET OPTION PUBLIC.max_temp_space = '1g';
```

Both of the following statements set a 1-MB limit for a connection:

```
SET OPTION PUBLIC.max_temp_space = 1048576;
```

```
SET OPTION PUBLIC.max_temp_space = '1m';
```

Use five percent of the total temporary space available:

```
SET OPTION PUBLIC.max_temp_space = '5p';
```

Related Information

temp_space_limit_check Option [page 845]
sa_disk_free_space System Procedure

1.5.6.7.83 min_password_length Option

Sets the minimum length for new passwords in the database.

Allowed Values

Integer

The value is in bytes. For single-byte character sets, this value is the same as the number of characters.

Default

6

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SECURITY OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SECURITY OPTION	No	No

Remarks

This option allows the database administrator to impose a minimum length on all new passwords for greater security. Existing passwords are not affected. Passwords have a maximum length of 255 bytes and are case sensitive.

Example

Set the minimum length for new passwords to 6 bytes.

```
SET OPTION PUBLIC.min_password_length = 6;
```

Related Information

Security: Passwords [page 1685]
verify_password_function Option [page 866]

1.5.6.7.84 min_role_admins Option

Sets the minimum number of administrators required to administer roles.

Allowed Values

Integer between 1 and 10.

Default

1

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SECURITY OPTION	No	No
Allowed to set temporarily?	No	No	No

Remarks

Setting this option takes effect immediately.

When creating, dropping, or revoking roles, changing a user password to null, locking users manually, or auto-locking a user for violating some condition of their login policy, the database server ensures that the number of role administrators will not drop below the specified minimum. Only role administrators with non-null passwords, including those with expired passwords, and who are not locked out are included in the count.

Changes to the value of this option are allowed only if each role has at least the specified number of role administrators. While revoking a role from a user, if the number of role administrators falls below the min_role_admins option value, the statement returns an error.

> **i** Note
>
> A user is locked manually by setting the *locked* attribute to ON for the login policy of that user. This prevents the user from making a new connection to the database.
>
> Conditions in the login policy that can result in a user being auto-locked include exceeding max_failed_login_attempts or max_days_since_login.

Example

The following example sets the min_role_admins option to 2:

```
SET OPTION PUBLIC.MIN_ROLE_ADMINS = 2;
```

Related Information

Roles [page 1527]

1.5.6.7.85 ml_remote_id Option [MobiLink]

Sets the remote ID for a remote database in a MobiLink synchronization system.

Allowed Values

Any value that uniquely identifies the database for MobiLink synchronization.

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

MobiLink synchronization uses the PUBLIC role setting only.

This option assigns a remote ID for a remote database in a MobiLink synchronization system. The remote ID must be unique within your synchronization system.

If you set the remote ID manually and you subsequently recreate the remote database, you must either give the recreated remote database a different name from the old one or use the ml_reset_sync_state stored procedure to reset the state information in the consolidated database for the remote database.

> ⚠ Caution
>
> The safest time to change the remote ID is before the first synchronization. If you change it later, be sure you have performed a complete, successful synchronization just before changing the remote ID. Otherwise you may lose data and put your database into an inconsistent state.

Example

The following SQL statement sets the remote ID to the value HR001:

```
SET OPTION PUBLIC.ml_remote_id = 'HR001'
```

Related Information

Remote IDs
MobiLink Remote ID Name
Remote IDs and MobiLink User Names in Scripts
ml_reset_sync_state System Procedure

1.5.6.7.86 nearest_century Option

Controls the interpretation of two-digit years in string-to-date conversions.

Allowed Values

Integer, between 0 and 100 inclusive

Default

50

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option controls the handling of two-digit years when converting from strings to dates or timestamps.

The nearest_century setting is a numeric value that acts as a rollover point. Two-digit years less than the value are converted to 20yy, while years greater than or equal to the value are converted to 19yy.

The historical SQL Anywhere behavior is to add 1900 to the year. Adaptive Server Enterprise behavior is to use the nearest century, so for any year where value `yy` is less than 50, the year is set to 20yy.

Related Information

Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]

1.5.6.7.87 non_keywords Option

Disables individual reserved keywords, allowing their use as identifiers.

Allowed Values

String

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

Specify a comma-separated list of keywords to prevent them from being recognized as reserved keywords. This option provides a way of ensuring that applications created with older versions of the product are not broken by the addition of new keywords. If you have an identifier in your database that is now a keyword, you can either enclose all references to it with double quotes, square brackets, or back quotes (`...`), or disable the keyword using the non_keywords option.

Whether a word is acceptable in the comma-separated list of words is determined by the following rules:

- It must appear in the list of reserved word candidates.
- It must not be one of the reserved words SET, OPTION, or OPTIONS.

The list of reserved word candidates is produced by the sa_reserved_words system procedure. The terms "reserved word" and "reserved keyword" are used interchangeably in this documentation.

The following statement prevents TRUNCATE and SYNCHRONIZE from being recognized as keywords:

```
SET OPTION non_keywords = 'TRUNCATE, SYNCHRONIZE';
```

Each new setting of this option replaces the previous setting.

The following statement clears all previous settings.

```
SET OPTION non_keywords =;
```

A side-effect of this option is that SQL statements that require a disabled keyword cannot be executed: they produce a syntax error. For example, if you disable the SELECT keyword, then SELECT statements cannot be executed.

You can enable individual keywords using the reserved_keywords option.

Related Information

Keywords
Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
reserved_keywords Option [page 813]
reserved_keywords Option [page 813]

1.5.6.7.88 odbc_describe_binary_as_varbinary Option

Controls how the SQL Anywhere ODBC driver describes BINARY columns.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option allows you to choose whether you want all BINARY and VARBINARY columns to be described to your application as BINARY or VARBINARY. By default, the SQL Anywhere ODBC driver describes both BINARY and VARBINARY columns as SQL_BINARY. When this option is set to On, the ODBC driver describes BINARY and VARBINARY columns as SQL_VARBINARY. Regardless of the setting of this option, it is not possible to distinguish between BINARY and VARBINARY columns.

It may be useful to turn this option On if you are using Delphi applications where BINARY columns are always zero-padded, but VARBINARY columns are not. You can improve performance in Delphi by setting this option to On so that all columns are treated as variable length data types.

Related Information

BINARY Data Type
VARBINARY Data Type

1.5.6.7.89 odbc_distinguish_char_and_varchar Option

Controls how the SQL Anywhere ODBC driver describes CHAR columns.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

When a connection is opened, the SQL Anywhere ODBC driver uses the setting of this option to determine how CHAR columns are described. If this option is set to Off (the default), then CHAR columns are described as SQL_VARCHAR. If this option is set to On, then CHAR columns are described as SQL_CHAR. VARCHAR columns are always described as SQL_VARCHAR.

Related Information

CHAR Data Type
VARCHAR Data Type

1.5.6.7.90 oem_string Option

Stores user-specified information in the header page of the database file.

Allowed Values

String (up to 128 bytes)

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

You can store information in the header page of the database file and later extract the information by reading the file directly from your application. This page is stored in the system dbspace file header. If you specify a value for the OEM string that is longer than 128 bytes, an error is returned.

You may find it useful to store such information as schema versions, the application name, the application version, and so on. Alternatively, without starting the database, an application could use the OEM string to determine whether the database file is associated with the application, or design your application to use the information to validate that the database file is intended for your application by storing a string that the application reads for validation purposes before using the database file. You could also extract metadata to display to users.

To set the oem_string in the system dbspace file header, execute the following statement:

```
SET OPTION PUBLIC.oem_string=user-specified-string;
```

The `user-specified-string` value is stored both in the ISYSOPTION system table and the system dbspace file header. You must define the string in the required character set before you specify it in a SET OPTION statement because no translation is done on the string when it is supplied in the SET OPTION statement. You can use the CSCONVERT function to convert the string to the required character set.

You can query the value of the oem_string in the following ways:

- Using the oem_string connection property:

  ```
  SELECT CONNECTION_PROPERTY( 'oem_string' );
  ```

- Using the SYSOPTION system view:

  ```
  SELECT setting FROM SYSOPTION WHERE "option" = 'oem_string';
  ```

Two sample programs in the `oem_string` directory are included:

- `dboem.cpp` is a C program that illustrates how to extract the OEM string and print it to the database server messages window.
- `dboem.pl` illustrates how to extract the OEM string and print it to the stdout within a PERL script.

> ⚠ Caution
>
> Applications cannot write directly to the OEM string in the database because it corrupts the database header page.

On Microsoft Windows, applications cannot read the file directly when a server has the database file loaded. The database server has an exclusive lock on the file. However, on any supported UNIX or Linux platform, applications that have read permissions can read the file directly at any time. However, changes to the OEM string may not show up in the file immediately. Issuing a checkpoint causes the database server to flush page 0 to disk, and reflect the current OEM string value.

Should the database server fail between changing the OEM string and the next checkpoint, the file header may not reflect the new OEM string value; the new OEM string value is set correctly after the database goes through recovery.

Example

The following example encrypts the OEM string that contains information about the database file and stores it in the database header file:

```
BEGIN
  DECLARE @v VARCHAR(100);
  SET @v = BASE64_ENCODE( ENCRYPT( 'database version 10', 'abc' ) );
  EXECUTE IMMEDIATE 'SET OPTION PUBLIC.oem_string = ''' || @v || '''';
END;
```

You can retrieve the value of the OEM string by executing the following statement:

```
SELECT DECRYPT(
   BASE64_DECODE(
      CONNECTION_PROPERTY( 'oem_string' ) ),'abc' )
```

Related Information

CSCONVERT Function [String]

1.5.6.7.91 on_charset_conversion_failure Option

Controls what happens if an error is encountered during character conversion.

Allowed Values

Ignore, Warning, Error

Default

Ignore

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

Controls what happens if an error is encountered during character conversion, as follows:

Ignore

Errors and warnings do not appear.

Warning

Reports substitutions and illegal characters as warnings. Illegal characters are not translated.

Error

Reports substitutions and illegal characters as errors.

When character set conversion is required between the client and the database, this option governs whether to ignore, return a warning, or return an error, when illegal characters are detected, or when character substitution is used.

Single byte to single byte converters are not able to report substitutions and illegal characters, and must be set to Ignore.

This option does not control the behavior when lossy conversion takes place on the client. For example, SQL statements from the client must be in, or converted to, the CHAR database character set. Suppose a Unicode client application prepares a SQL statement, and that statement contains characters that cannot be represented in the CHAR database character set. Substitution characters are used instead. However, because the lossy conversion took place on the client, the database server is unaware of the lossy conversion.

Related Information

Comparisons Between CHAR and NCHAR
NCHAR to CHAR Conversions
Lossy Conversion and Substitution Characters

1.5.6.7.92 on_tsql_error Option

Controls error-handling in stored procedures.

Allowed Values

Stop

Stop execution immediately upon finding an error.

Conditional

If the procedure uses ON EXCEPTION RESUME, and the statement following the error handles the error, continue, otherwise exit.

Continue

Continue execution, regardless of the following statement. If there are multiple errors and the last statement executed was not successful, then the first error encountered in the stored procedure is returned once the procedure completes.

Default

Conditional

A temporary setting of CONTINUE is established for the current user by SAP Open Client and jConnect TDS connections (including the Adaptive Server Enterprise isql tool).

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option controls error handling in stored procedures and Transact-SQL batches. This option has no effect within the TRY block of a BEGIN...END statement, and the error handling behavior defined within the TRY block.

Both the Conditional and Continue settings for on_tsql_error are used for Adaptive Server Enterprise compatibility, with Continue most closely simulating Adaptive Server Enterprise behavior. To have errors reported earlier, use the Conditional setting when creating new Transact-SQL stored procedures.

When this option is set to Stop or Continue, it supersedes the setting of the continue_after_raiserror option. However, when this option is set to Conditional (the default), behavior following a RAISERROR statement is determined by the setting of the continue_after_raiserror option.

Related Information

Error Handling with ON EXCEPTION RESUME
Transact-SQL Procedure Language
Open Client/jConnect TDS Compatibility Options [page 874]
Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
continue_after_raiserror Option [page 710]
sp_tsql_environment System Procedure

1.5.6.7.93 optimization_goal Option

Determines whether query processing is optimized towards returning the first row quickly, or minimizing the cost of returning the complete result set.

Allowed Values

First-row, All-rows

Default

All-rows

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION

	PUBLIC role	For current user	For other users
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The optimization_goal option controls whether SQL data manipulation language (DML) statements are optimized for response time or total resource consumption.

If the option is set to All-rows (the default), then a query is optimized to choose an access plan with the minimal estimated total retrieval time. Setting optimization_goal to All-rows may be appropriate for applications that intend to process the entire result set, such as SAP PowerBuilder DataWindow applications. A setting of All-rows is also appropriate for insensitive (ODBC static) cursors since the entire result is materialized when the cursor is opened. It may also be appropriate for scroll (ODBC keyset-driven) cursors, since the intent of such a cursor is to permit scrolling through the result set.

If the option is set to First-row, the optimizer chooses an access plan that is intended to reduce the time to fetch the first row of the query's result, possibly at the expense of total retrieval time. In particular, the optimizer typically avoids, if possible, access plans that require the materialization of results to reduce the time to return the first row. With this setting, the optimizer favors access plans that use an index to satisfy a query's ORDER BY clause, rather than plans that require an explicit sorting operation.

You can use the FASTFIRSTROW table hint in a query's FROM clause to set the optimization goal for a specific query to First-row, without having to change the optimization_goal setting.

You can override any temporary or PUBLIC settings for this option within individual INSERT, UPDATE, DELETE, SELECT, UNION, EXCEPT, and INTERSECT statements by including an OPTION clause in the statement.

Related Information

FROM Clause

1.5.6.7.94 optimization_level Option

Controls the amount of effort made by the query optimizer to find an access plan for a SQL statement.

Allowed Values

0-15

Default

9

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The optimization_level option controls the amount of effort that the optimizer spends on optimizing SQL data manipulation language (DML) statements. This option controls the maximum number of alternative join strategies that the optimizer considers for any SELECT block. The higher the setting of optimization_level, the greater the maximum number of join strategies that the optimizer considers.

If the option is set to 0, then the optimizer chooses the first access plan it considers for execution, in effect avoiding any cost-based comparison of alternative plans. In addition, with level 0 some semantic optimizations of nested queries are disabled. If this option is set to a value higher than 0, the optimizer evaluates alternative strategies and chooses the one with the lowest expected cost. If this option is set to a value greater than the default (9), the optimizer is more aggressive in its search for alternative strategies, possibly resulting in much higher elapsed time spent in the optimization phase.

In typical scenarios, this option is temporarily set to lower levels (0, 1, or 2) when the application desires faster OPEN times for a DML statement. It is known that although the statement may be complex, the query's execution time is very small, and the specific access plan chosen by the optimizer is less consequential. It is not recommended that the PUBLIC setting of optimization_level be changed from its default.

The effect of setting the optimization_level option is independent of the settings of the optimization_goal and optimization_workload options.

Simple DML statements (single-block, single-table queries that contain equality conditions in the WHERE clause that uniquely identify a specific row) are optimized heuristically and bypass the cost-based optimizer altogether. The optimization of simple DML statements is not affected by the setting of the optimization_level option. The count of the number of requests optimized through the optimizer bypass mechanism is available as the QueryBypassed connection property.

You can override any temporary or PUBLIC settings for this option within individual INSERT, UPDATE, DELETE, SELECT, UNION, EXCEPT, and INTERSECT statements by including an OPTION clause in the statement.

Related Information

How the Optimizer Works
List of Connection Properties [page 880]

1.5.6.7.95 optimization_workload Option

Determines whether query processing is optimized towards a workload that is a mix of updates and reads or a workload that is predominantly read-based (OLAP).

Allowed Values

Mixed, OLAP

Default

Mixed

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

The optimization_workload option controls whether queries are optimized for a workload that is a mix of updates and reads or that is predominantly read-only.

If the option is set to Mixed (the default), the optimizer chooses query optimization algorithms appropriate for a workload that is a mixture of short inserts, updates, and deletes and longer running read-only queries.

If the option is set to OLAP, the optimizer chooses algorithms appropriate for a workload that consists for the most part of long-running queries, combined with batch updates. In particular, the optimizer may choose to use the Clustered Hash Group By query execution algorithm.

When the option is set to OLAP, the Clustered Hash Group By algorithm is enabled. If the option is set to Mixed (the default), it is disabled.

You can override any temporary or PUBLIC settings for this option within individual INSERT, UPDATE, DELETE, SELECT, UNION, EXCEPT, and INTERSECT statements by including an OPTION clause in the statement.

Related Information

optimization_level Option [page 786]

1.5.6.7.96 parameterization_level Option

Controls automatic parameterization of client statements.

Allowed Values

Off, Simple, Forced

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

When a client application prepares a SQL statement, the database server may choose to replace constant literals within the SQL text with parameter placeholders prior to preparing the statement. The resulting SQL statement is more general. When subsequent SQL statements that differ only by values of the parameterized

constants are executed, they can be matched with the parameterized statement. Generalized SQL statements enable two potential performance benefits:

- If the parameterized statement is cached by the client connection, then the preparation of future SQL statements may be avoided.
- If the database server caches an execution plan for the parameterized statement, then optimization of future SQL statements may be avoided.

The first benefit requires client statement caching to be enabled, while the second benefit requires both client statement caching and server plan caching to be enabled.

Parameterization is transparent to the client application, and automatically inserted parameters are not visible to a DESCRIBE of the statement or cursor.

Parameterization behavior is controlled by the following values for this option:

Off Statements are not parameterized. This setting corresponds to behavior in versions 16 and earlier of the software.

Simple The database server decides when to parameterize statements and which constants within each statement to parameterize.

Forced The database server parameterizes every statement as soon as possible and all eligible constants within each statement are replaced with placeholders. However, there are some statement types and query constructs for which parameterization is not supported either because it is syntactically invalid or because it is not generally useful, so constant literals within those contexts are not replaced.

You can override any temporary or PUBLIC settings for this option within individual INSERT, UPDATE, DELETE, and SELECT statements by including an OPTION clause in the statement.

Related Information

Plan Caching
Plan Caching
sp_plancache_contents System Procedure
max_client_statements_cached Option [page 759]
List of Connection Properties [page 880]

1.5.6.7.97 pinned_cursor_percent_of_cache Option

Specifies how much of the cache can be used for pinning cursors.

Allowed Values

Integer, between 0-100

Default

10

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

The database server uses pages of virtual memory for the data structures needed to implement cursors. These pages are kept locked in memory between fetch requests so they are readily available when the next fetch request arrives.

To prevent these pages from occupying too much of the cache in low memory environments, a limit is placed on the percentage of the cache allowed to be used for pinning cursors. You can use the pinned_cursor_percent_of_cache option to adjust this limit.

The option value is specified as a percentage from 0 to 100, with a default of 10. Setting the option to 0 means that cursor pages are not pinned between fetch requests.

1.5.6.7.98 post_login_procedure Option

Specifies a procedure whose result set contains messages that should be displayed by applications when a user connects.

Allowed Values

String

Default

dbo.sa_post_login_procedure

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SECURITY OPTION	Yes, with SET ANY SECURITY OPTION	Yes, with SET ANY SECURITY OPTION
Allowed to set temporarily?	Yes, with SET ANY SECURITY OPTION	Yes (current connection only), with SET ANY SECURITY OPTION	No

Remarks

When the post_login_procedure option is set to anything other than an empty string, applications can call the procedure specified by the option as part of the connection process to determine what messages should be displayed to the user, if any. The option value should be of the form **owner.function-name** to prevent a user from overriding the function.

The SQL Anywhere 17 plug-in for SQL Central and Interactive SQL call the procedure if this option is set and display any messages returned by the procedure in a window. Other applications that are not included with SQL Anywhere should be modified to call the procedure given by this option and display messages if you need this functionality.

One case where an application might display a message on connection is to notify the user that their password is about to expire if a password expiry system is implemented. This functionality could be used to notify the user each time they connect if their password will expire in the next few days, and before it actually expires.

The procedure specified by this option must return a result set with one or more rows and two columns. The first column of type VARCHAR(255) returns the text of the message, or NULL if there is no message. The second column of type INT returns the action type. Allowed values for actions are:

0

Display the message (if any).

1

Display the message and prompt the user for a password change.

2-99

Reserved.

100 and greater

User defined.

The SQL Anywhere 17 plug-in and Interactive SQL (dbisql) display all non-NULL messages, regardless of the action value. If the action is set to 1, then these tools prompt the user to change the password, and then set the new password to the user-specified value.

For an example that uses post_login_procedure and includes advanced password rules and implementing password expiration, see Using a password verification function.

Example

The following example uses a procedure named p_post_login_check that warns users that their password is about to expire and then prompts them to change their password.

```
CREATE PROCEDURE DBA.p_post_login_check( )
RESULT( message_text VARCHAR(255), message_action INT )
BEGIN
  DECLARE message_text        CHAR(255);
  DECLARE message_action     INT;

  -- assume the password_about_to_expire variable was
  -- set by the login procedure
  IF password_about_to_expire = 1 THEN
    SET message_text = 'Your password is about to expire';
    SET message_action = 1;
  ELSE
    SET message_text = NULL;
    SET message_action = 0;
  END IF;
  -- return message (if any) through this result set
  SELECT message_text, message_action;
END;
GRANT EXECUTE ON DBA.p_post_login_check TO PUBLIC;
SET OPTION PUBLIC.post_login_procedure = 'DBA.p_post_login_check';
```

Related Information

Security: Passwords [page 1685]
login_procedure Option [page 755]
verify_password_function Option [page 866]
NewPassword (NEWPWD) Connection Parameter [page 98]
sa_post_login_procedure System Procedure

1.5.6.7.99 precision Option

Specifies the maximum number of digits in the result of any decimal arithmetic.

Allowed Values

Integer, between 1 and 127, inclusive

Default

30

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	No	No
Allowed to set temporarily?	No	No	No

Remarks

Precision is the total number of digits in a number and does not include the decimal point. The scale option specifies the minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum precision. For example, the number 3.1415926 has a precision of 8 and a scale of 7.

Multiplication, division, addition, subtraction, and aggregate functions can all have results that exceed the maximum precision.

For example, when a DECIMAL(8,2) is multiplied with a DECIMAL(9,2), the result could require a DECIMAL(17,4). If precision is 15, only 15 digits are kept in the result. If scale is 4, the result is a DECIMAL(15,4). If scale is 2, the result is a DECIMAL(15,2). In both cases, there is a possibility of overflow.

Related Information

scale Option [page 818]
DECIMAL Data Type
NUMERIC Data Type

1.5.6.7.100 prefetch Option

Controls whether rows are fetched to the client side before being made available to the client application.

Allowed Values

Off, Conditional, Always

Default

Conditional

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option controls whether rows are fetched to the client side in advance of being made available to the client application. Fetching several rows at a time, even when the client application requests rows one at a time (for example, when looping over the rows of a cursor), can cut down on response time and improve overall throughput by cutting down the number of requests to the database.

- Off means that no prefetching is done.
- Conditional (the default) causes prefetching to occur unless the cursor type is SENSITIVE or the query includes a proxy table.
- Always means that prefetching is done even for sensitive cursor types and cursors that involve a proxy table.

The Always value must be used with caution, as it affects some cursor semantics. For example, it causes the sensitive cursor to become asensitive. Old values may be fetched if the value was updated between the prefetch and the application's fetch request. In addition, using prefetch on a cursor that involves a proxy table can cause the error -668, Cursor is restricted to FETCH NEXT operations, if the client attempts to re-fetch prefetch rows. A client may attempt to re-fetch prefetch rows after a rollback or on a fetch relative 0, if a fetch column is re-bound or bound for the first time after the first fetch, or when GET DATA is used.

The value sensitive cursor types include the ESQL SENSITIVE and SCROLL cursor types, and the ODBC and OLE DB DYNAMIC and KEYSET cursor types.

The setting of the prefetch option is ignored by SAP Open Client and jConnect connections.

If the DisableMultiRowFetch connection parameter is set to YES, the prefetch database option is ignored and no prefetching is done.

This option previously accepted the value On. This value is now an alias for Conditional.

Related Information

Prefetches
DisableMultiRowFetch (DMRF) Connection Parameter [page 76]

1.5.6.7.101 preserve_source_format Option

Controls whether the original source definition of procedures, triggers, views, and event handlers is saved in system files.

If saved, it is saved in the column source in SYSTAB, SYSPROCEDURE, SYSTRIGGER, and SYSEVENT.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

When preserve_source_format is On, the database server saves the formatted source from CREATE and ALTER statements on procedures, views, triggers, and events, and puts it in the appropriate system view's source column.

Unformatted source text is stored in the same system tables, in the columns proc_defn, trigger_defn, and view_defn. However, these definitions are not easy to read in SQL Central. The formatted source column allows you to view the definitions with the spacing, comments, and case that you want.

This option can be turned off to reduce space used to save object definitions in the database.

1.5.6.7.102 prevent_article_pkey_update Option [MobiLink]

Controls updates to the primary key columns of tables involved in MobiLink publications.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

Setting this option to On disallows updates to the primary key columns of tables that are part of a publication. This option helps ensure data integrity, especially in a replication and synchronization environment.

> ⚠ Caution
>
> It is strongly recommended that you do not set this option to Off in a synchronization or replication environment.

1.5.6.7.103 priority Option

Sets the priority level at which requests from a connection are executed.

Allowed Values

Critical, High, Above Normal, Normal, Below Normal, Low, Background

Default

normal

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

Setting of this option takes effect immediately.

The value of this option cannot be set higher than the value of the max_priority option.

Related Information

max_priority Option [page 766]

1.5.6.7.104 progress_messages Option

Controls whether progress messages are sent from the database server to the client.

Allowed Values

Off

Progress messages are not sent to the client.

Raw

When Raw is selected, the following format is used for progress messages:

```
43;9728;22230;pages;5025;6138
```

Raw progress messages have six fields separated by semicolons that are defined as follows:

Field 1

The percentage of statement executed.

Field 2

The number of completed pages, rows, or bytes.

Field 3

The number of pages, rows, or bytes to be processed.

Field 4

What is being processed: pages, rows, or bytes.

Field 5

The current elapsed time displayed in milliseconds.

Field 6

The estimated time in milliseconds remaining to complete the execution of the statement.

Formatted

When Formatted is selected, the following format is used for progress messages:

```
43% (9728 of 22230 pages) complete after 00:00:05; estimated 00:00:06
remaining
```

Formatted progress messages are localized, and the time format is HH:MM:SS. Units less than 100 KB are displayed in bytes, units less than 100 MB are displayed in KB, and units greater than 100 MB are displayed in MB.

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

Progress messages are sent at intervals that are 5% of the total estimated duration of the statement. Typically, the estimate is completed and the first progress message is sent within 10 seconds. Additional progress messages are sent in intervals of 30 seconds to 5 minutes. If the percentage complete is identical to the value sent in a previous message, an updated progress message is not sent until more than 5 minutes have elapsed since the last message was sent. Progress messages are not sent for statements that take less than 30 seconds to execute.

Estimates are recalculated continually; the accuracy of the remaining time estimate increases as the operation progresses. During events such as backups, the total number of pages may be adjusted during statement execution, so the percent complete and remaining time estimates change. With statements such as BACKUP...WITH CHECKPOINT COPY or UNLOAD SELECT the total number of affected pages or rows is unknown and it is possible for the percentage complete value to be greater than 100%. As a result, the estimated remaining time cannot be calculated and it is not included in the progress message.

The following statements and procedures support progress messages:

- BACKUP DATABASE (both image and archive)
- LOAD TABLE (USING FILE and USING CLIENT FILE only)
- MESSAGE
- REORGANIZE TABLE
- RESTORE DATABASE
- UNLOAD (all types)
- sa_table_page_usage system procedure

You can set the progress_messages option when you are connected to the utility database using the SET OPTION statement.

You can also set the progress_messages option in Interactive SQL by clicking *Tools* > *Options* > *SQL Anywhere* > *Commands*, and then clicking *Show Progress Messages*. When *Show Progress Messages* is enabled, the progress_messages option is set to Formatted.

Related Information

The Utility Database (utility_db) [page 309]
SET OPTION Statement

Backup Utility (dbbackup) [page 1121]
sa_table_page_usage System Procedure

1.5.6.7.105 qualify_owners Option [SQL Remote]

Controls whether SQL statements being replicated by SQL Remote should use qualified object names.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

When qualification is not needed in SQL Anywhere installations, messages are slightly smaller with this option set to Off.

Related Information

SQL Remote Options

1.5.6.7.106 query_mem_timeout Option

Sets the maximum time, in milliseconds, that a request waits for a memory grant.

Allowed Values

-1, 0, positive integer

Default

-1

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

When this option is set to -1 (the default) or any value less than 0, the request waits for a memory grant for up to 50 times the estimated execution time for the request. If this option is set to 0, the request waits forever for memory to be granted. Otherwise, the value is the maximum time, in milliseconds, that a request waits for a memory grant.

Related Information

Cache and the Memory Governor [page 1441]

1.5.6.7.107 quote_all_identifiers Option [SQL Remote]

Controls whether SQL statements being replicated by SQL Remote should use quoted identifiers.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

When this option is Off, dbremote quotes identifiers that require quotes by SQL Anywhere.

When the option is On, all identifiers are quoted.

Related Information

SQL Remote Options

1.5.6.7.108 quoted_identifier Option

Controls the interpretation of strings that are enclosed in double quotes.

Allowed Values

On, Off

Default

On

A temporary setting for the current user is established by SAP Open Client and jConnect TDS connections ('Off').

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option controls whether strings that are enclosed in double quotes are interpreted as identifiers (On) or as literal strings (Off). The quoted_identifier option is included for Transact-SQL compatibility.

Related Information

Open Client/jConnect TDS Compatibility Options [page 874]
Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
Options for Transact-SQL Compatibility
sp_tsql_environment System Procedure
SET Statement [T-SQL]

1.5.6.7.109 read_past_deleted Option

Controls database server behavior on uncommitted deletes at isolation levels 1 and 2.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

If read_past_deleted is On (the default), sequential scans at isolation levels 1 and 2 skip uncommitted deleted rows. If Off, sequential scans block on uncommitted deleted rows at isolation levels 1 and 2 (until the deleting transaction commits or rolls back). This option changes server behavior at isolation levels 1 and 2.

For most purposes, this option should be left On. If set to Off, the blocking behavior depends on the plan chosen by the optimizer (if there is an index that could possibly be used).

Related Information

Isolation Levels and Consistency

1.5.6.7.110 recovery_time Option

Sets the maximum length of time, in minutes, that the database server takes to recover from system failure.

Allowed Values

Integer, in minutes

Default

2

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

You may need to restart the database server for changes to this option setting to take effect.

This option is used with the checkpoint_time option to decide when checkpoints should be done.

The database server uses a heuristic to estimate the recovery time based on the operations that have been performed since the last checkpoint, and includes both the estimated recovery time and the estimated checkpoint time for the database. So, the recovery time is not exact.

Related Information

The Automatic Recovery Process [page 962]
How the Database Server Decides When to Checkpoint [page 306]
checkpoint_time Option [page 700]
-gr Database Server Option [page 444]

1.5.6.7.111 remote_idle_timeout Option

Controls how many seconds of inactivity web service client procedures and functions tolerate.

Allowed Values

Integer, in seconds

Default

15

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option affects web service client procedures and functions. If more time than the specified number of seconds passes without activity, the procedure or function times out.

Related Information

Web Client Application Development
CREATE PROCEDURE Statement [Web Service]
CREATE FUNCTION Statement [Web Service]

1.5.6.7.112 replication_error Option [SQL Remote]

Allows you to specify a stored procedure to be called by the Message Agent when a SQL error occurs.

Allowed Values

Stored procedure name

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

For SQL Remote, the replication_error option allows you to specify a stored procedure to be called by the Message Agent when a SQL error occurs. By default, no procedure is called.

The procedure must have a single argument of type CHAR, VARCHAR, or LONG VARCHAR. The procedure is called once with the SQL error message and once with the SQL statement that causes the error. In some circumstances (such as foreign key violations), the SQL statement that caused the error is not available, so the stored procedure can only be called once.

Although the option allows you to track and monitor SQL errors in replication, you must still design them out of your setup; this option is not intended to resolve such errors.

Related Information

Replication Error-handling Procedures
SQL Remote Options
replication_error_piece Option [SQL Remote] [page 809]

1.5.6.7.113 replication_error_piece Option [SQL Remote]

Works with the replication_error option to allow you to specify a LONG VARCHAR stored procedure to be called by the Message Agent when a SQL error occurs during SQL Remote replication.

Allowed Values

Stored procedure name

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

If an error occurs and replication_error is defined, then the replication_error procedure is called with the full error string.

If replication_error and replication_error_piece are both defined, then the error is broken up into VARCHAR pieces. replication_error is called with the first piece and replication_error_piece is called repeatedly with the remaining pieces.

Related Information

replication_error Option [SQL Remote] [page 808]
SQL Remote Options

1.5.6.7.114 request_timeout Option

Controls the maximum time a single request can run. This option can be used to prevent a connection from consuming a significant amount of server resources for a long period of time.

Allowed Values

Integer, 0 through 86400 (one day), in seconds

Default

0

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

When this option is set to 0, requests do not time out.

Any request that takes longer than approximately request_timeout seconds (wall-clock time, not CPU time) is interrupted and an error is returned to the user. The error returned is SQLE_REQUEST_TIMEOUT: "Request interrupted due to timeout". If a request is blocked, and the blocking_timeout option is set to 0, then the request can remain blocked for a maximum of request_timeout seconds before returning a blocking error (for example, SQLE_LOCKED: "User '%1' has the row in '%2' locked").

Values in the range 1 to 14 cannot be specified as a user option or the PUBLIC option, but can be specified as a temporary option. This prevents users from being locked out of the database server if connecting takes a long time (for example, because of a complex login procedure).

This option can be used with both database client and HTTP/HTTPS requests. Setting the option in a stored procedure or HTTP/HTTPS request has no effect on the current request since the option value at the beginning of the request is used.

Setting the request_timeout public option should be done with caution as this can cause applications that have long running requests (such as dbvalid, dbbackup, and dbunload) to fail. Also, applications that do not use

significant server resources, but that can block on another user can fail when request_timeout is set. One way to address these types of problems is to set the request_timeout option only for certain applications in the login procedure based on a connection's APPINFO value.

Setting this option may not prevent applications from using significant server resources if each request evaluates quickly, for example when fetching a result set containing many rows.

Related Information

blocking_timeout Option [page 697]
AppInfo (APP) Connection Parameter [page 49]

1.5.6.7.115 RESERVED_CONNECTIONS Option

Controls the minimum number of concurrent connections that a database must reserve for standard connections. This option is useful when your database server accepts HTTP/HTTPS connections.

Allowed Values

size

This integer specifies the number of concurrent connections that the database must reserve for standard connections.

For databases running on the network database server, specify a number that fulfills the following requirements:

- Less than the maximum database server connection limit as allowed by your license.
- Less than the maximum database server connection limit as specified by the -gm database server option.
- Less than the maximum database connection limit specified by the max_connections database option.

For the personal database server, specify a number between 3 and 9 .The personal database server (dbeng17) accepts a maximum of 10 concurrent connections, of which a minimum of 3 are always reserved for standard connections.

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes with SET ANY SYSTEM OPTION	No	No

	PUBLIC role	For current user	For other users
Allowed to set temporarily?	Yes with SET ANY SYSTEM OPTION	No	No

Remarks

Setting this option ensures that a database can accept standard connections even when its HTTP/HTTPS connections are queued.

A database server accepts HTTP/HTTPS connections until it reaches its license limit, and then it queues subsequent HTTP/HTTPS connections and processes them as connections are made available. There is no opportunity for a standard connection to replace an HTTP/HTTPS connection while there are connection attempts in the queue. Users wanting to make a standard connection, could wait indefinitely for the HTTP/HTTPS connection queue to complete. Use the reserved_connections option to specify a minimum number of database connections that can only accept standard connections.

The personal database server always reserves a minimum of 3 connections for standard connections to ensure that standard database connections are not blocked by HTTP connections.

By default, the network database server does not reserve connections.

You can view the current number of reserved connections for a database by querying the value of the reserved_connections connection property:

```
SELECT CONNECTION_PROPERTY ( 'reserved_connections' );
```

Example

The following example reserves 10 connections for standard connections.

```
SET OPTION PUBLIC.reserved_connections=10;
```

Related Information

Connection Limits for Databases and Database Servers [page 248]
Reserving Standard Connections [page 252]
SET OPTION Statement
max_connections Option [page 761]

1.5.6.7.116 reserved_keywords Option

Changes one or more keywords into reserved keywords.

Allowed Values

String

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

This option creates reserved keywords from specified keywords. Currently, only the keyword LIMIT can be enabled as a reserved keyword.

By default, LIMIT is not a reserved keyword so use of the LIMIT clause in a SELECT statement will result in a syntax error.

```
SELECT * FROM Products
    ORDER BY ID
    LIMIT 5;
```

This means that LIMIT could be used as an identifier in a CREATE TABLE statement for example. In order to enable support for the LIMIT clause, the keyword LIMIT must be enabled as a reserved keyword using the reserved_keywords option. The following statement does this:

```
SET OPTION PUBLIC.reserved_keywords = 'LIMIT';
```

Once enabled as a reserved keyword, LIMIT cannot be used as an identifier unless it is enclosed by double quotes, square brackets, or back quotes (`...`).

```
SELECT * FROM [LIMIT];
```

Whether a word is acceptable in the comma-separated list of words is determined by the following rules:

- It must appear in the list of reserved word candidates.
- It must not be one of the reserved words SET, OPTION, or OPTIONS.

The list of reserved word candidates is produced by the sa_reserved_words system procedure. The terms "reserved word" and "reserved keyword" are used interchangeably in this documentation.

You can disable individual reserved keywords using the non_keywords option.

Adding a keyword to the reserved_keywords list that is already a reserved keyword has no effect.

Adding a keyword to the reserved_keywords list that has already been added to the non_keywords list has no effect.

Each new setting of this option replaces the previous setting.

The following statement clears all previous settings.

```
SET OPTION PUBLIC.reserved_keywords =;
```

Related Information

Keywords
Reserved Words
non_keywords Option [page 777]

1.5.6.7.117 return_date_time_as_string Option

Controls how a DATE, TIME, or TIMESTAMP value is passed to the client application when queried.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	No	No	No
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option indicates whether DATE, TIME, and TIMESTAMP values are returned to applications as a DATE, TIME, or TIMESTAMP data type or as a string.

When this option is set to On, the database server converts the DATE, TIME, or TIMESTAMP value to a string before it is sent to the client to preserve the date_format, time_format, or timestamp_format option setting.

SQL Central and Interactive SQL automatically turn the return_date_time_as_string option Off. When this option is set Off, then Interactive SQL displays DATE, TIME, and TIMESTAMP values using the locale settings of the client computer.

Related Information

date_format Option [page 716]
time_format Option [page 847]
timestamp_format Option [page 850]
timestamp_with_time_zone_format Option [page 852]

1.5.6.7.118 rollback_on_deadlock Option

Controls how transactions are treated when a deadlock occurs.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

When this option is set to On, a transaction is automatically rolled back if it encounters a deadlock. The rollback happens after the current request completes. If this option is set to Off, the database server automatically rolls back the statement that encountered the deadlock, and returns an error to that transaction indicating which form of deadlock occurred. Rolling back the statement likely would not release any of the locks acquired by the statement.

Related Information

Deadlocks
ICU - International Components for Unicode

1.5.6.7.119 row_counts Option

Specifies whether the database always count the number of rows in a query when it is opened.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

If this option is set to Off, the row count is usually only an estimate. If this option is set to On, the row count is always accurate.

> ⚠ Caution
>
> When row_counts is set to On, it may take significantly longer to execute queries. In fact, it usually causes the database server to execute the query twice, doubling the execution time.

Related Information

sasql_num_rows

1.5.6.7.120 save_remote_passwords Option [SQL Remote]

Saves the password that is entered in the message link.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

If you are storing the message link parameters externally, rather than in the database, you may not want to save the passwords. You can prevent the passwords from being saved by setting this option to Off.

Related Information

SQL Remote Options

1.5.6.7.121 scale Option

Specifies the minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum precision.

Allowed Values

Integer, between 0 and 127, inclusive, and less than the value specified for the precision database option

Default

6

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	No	No
Allowed to set temporarily?	No	No	No

Remarks

Precision is the total number of digits in a number and does not include the decimal point. The scale option specifies the minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum precision. For example, the number 3.1415926 has a precision of 8 and a scale of 7.

Multiplication, division, addition, subtraction, and aggregate functions can all have results that exceed the maximum precision.

Related Information

precision Option [page 793]

1.5.6.7.122 secure_feature_key Option [Deprecated]

Allows you to enable features for the connection that were secured using the -sf database option.

Allowed Values

String

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	No	No	No
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

You can specify features that cannot be used by databases running on a server by including the -sf database server option when you start the database server. The -sk server option lets you specify a key that can be used to re-enable all secured (disabled) features for a connection and gives that connection permission to change the features that are secured for all databases running on the database server. When you set the value of the secure_feature_key temporary option to the value specified by -sk when the database server was started, then all features are re-enabled for that database connection, and on that connection you can use the sa_server_option system procedure to control access to database features.

If the secure_feature_key option is set to any value other than the one specified by the -sk database server option, no error is given, and the features specified by -sf database server option remain disabled for the connection.

Example

The following command starts a database server named secure_server with access to the request log and all remote data access features disabled. The key specified by the -sk option can be used later to enable these features for a specific database connection.

```
dbsrv17 -n secure_server -sf request_log,remote -sk j978kls12 testdb.db
```

Setting the secure_feature_key option to the value specified by -sk for a database running on the secure_server database server enables access to the request log and remote data access features for that connection:

```
SET TEMPORARY OPTION secure_feature_key = 'j978kls12';
```

Related Information

Creating Secured Feature Keys [page 1690]
sp_use_secure_feature_key System Procedure
-sk Database Server Option [page 500]
-sf Database Server Option [page 493]
sa_server_option System Procedure

1.5.6.7.123 sort_collation Option

Allows implicit use of the SORTKEY function on ORDER BY expressions.

Allowed Values

Internal, `collation-name`, or `collation-id`

Default

Internal

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

When the value of this option is Internal, the ORDER BY clause is treated as is.

When the value of this option is set to a valid collation name or collation ID, CHAR or NCHAR string expressions in the ORDER BY clause are treated as if the SORTKEY function had been invoked. String expressions that use other string data types, such as BINARY, UUID, XML, or VARBIT are not modified.

Example

Set the sort collation to binary:

```
SET TEMPORARY OPTION sort_collation='binary';
```

Having the sort collation set to binary transforms the following queries:

```
SELECT Name, ID
```

```
FROM Products
ORDER BY Name, ID;
SELECT name, ID
FROM Products
ORDER BY 1, 2;
```

The queries are transformed into:

```
SELECT Name, ID
FROM Products
ORDER BY SORTKEY(Name, 'binary'), ID;
```

Related Information

SORTKEY Function [String]

1.5.6.7.124 sql_flagger_error_level Option

Controls the response to any SQL that is not part of the specified standard.

Allowed Values

- Off
- SQL:1992/Entry
- SQL:1992/Intermediate
- SQL:1992/Full
- SQL:1999/Core
- SQL:1999/Package
- SQL:2003/Core
- SQL:2003/Package
- SQL:2008/Core
- SQL:2008/Package
- UltraLite

Default

W

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option flags any SQL that is not part of a specified standard as a warning. For example, specifying SQL:2003/Package causes the database server to flag syntax that is not full SQL/2003 syntax.

The default behavior, Off, turns error flagging off.

For compatibility with previous SQL Anywhere versions, the following values are also accepted:

E

This option corresponds to SQL:1992/Entry.

I

This option corresponds to SQL:1992/Intermediate.

F

This option corresponds to SQL:1992/Full.

W

This option corresponds with Off.

Related Information

Transactions-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]

sa_ansi_standard_packages System Procedure

SQLFLAGGER Function [Miscellaneous]

sql_flagger_warning_level Option [page 824]

The Embedded SQL Preprocessor

1.5.6.7.125 sql_flagger_warning_level Option

Controls the response to any SQL that is not part of the specified standard.

Allowed Values

- Off
- SQL:1992/Entry
- SQL:1992/Intermediate
- SQL:1992/Full
- SQL:1999/Core
- SQL:1999/Package
- SQL:2003/Core
- SQL:2003/Package
- SQL:2008/Core
- SQL:2008/Package
- UltraLite

Default

W

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option flags any SQL that is not part of a specified standard as a warning. For example, specifying SQL:2003/Package causes the database server to flag syntax that is not full SQL/2003 syntax.

The default behavior, Off, turns warning flagging off.

For compatibility with previous versions, the following values are also accepted:

E

This option corresponds to SQL:1992/Entry.

I

This option corresponds to SQL:1992/Intermediate.

F

This option corresponds to SQL:1992/Full.

W

This option corresponds to Off.

Related Information

Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
sa_ansi_standard_packages System Procedure
SQLFLAGGER Function [Miscellaneous]
sql_flagger_error_level Option [page 822]
The Embedded SQL Preprocessor

1.5.6.7.126 sr_date_format Option [SQL Remote]

Sets the format for DATE values that are retrieved from the database.

Allowed Values

String

Default

yyyy/mm/dd

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

The Message Agent uses this option when replicating columns that store a date. The format is a string using the following symbols:

Symbol	Description
YY	Two-digit year
YYYY	Four-digit year
MM	Two-digit month
MMM[m...]	Character short form for months (as many characters as there are "m"s)
DD	Two-digit day of month

Each symbol is substituted with the appropriate data for the date that is being formatted.

For symbols that represent character data (such as MMM), you can control the case of the output as follows:

- Type the symbol in all uppercase to have the format appear in all uppercase. For example, MMM produces JAN.
- Type the symbol in all lowercase to have the format appear in all lowercase. For example, mmm produces jan.
- Type the symbol in mixed case to have the database server choose the appropriate case for the language that is being used. For example, in English, typing Mmm produces May, while in French it produces mai.

If the character data is multibyte, the length of each symbol reflects the number of characters, not the number of bytes. For example, the 'MMM' symbol specifies a length of three characters for the month.

For symbols that represent numeric data, you can control zero-padding with the case of the symbols:

The option is a string build from the following symbols:

- Type the symbol in same-case (such as MM or mm) to allow zero padding. For example, yyyy/mm/dd could produce 2002/01/01.
- Type the symbol in mixed case (such as Mm) to suppress zero padding. For example, yyyy/Mm/Dd could produce 2002/1/1.

Related Information

sr_time_format Option [SQL Remote] [page 827]
sr_timestamp_format Option [SQL Remote] [page 828]
sr_timestamp_with_time_zone_format Option [SQL Remote] [page 830]
SQL Remote Options

1.5.6.7.127 sr_time_format Option [SQL Remote]

Sets the format for TIME values that are retrieved from the database.

Allowed Values

String

Default

hh:nn:ss.Ssssss

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

The Message Agent uses this option when replicating columns that store a time. The format is a string using the following symbols:

Symbol	Description
HH	Two-digit hours (24-hour clock).
NN	Two-digit minutes.

Symbol	Description
MM	Two-digit minutes if following a colon (as in HH:MM).
SS[.ssssss]	Seconds and fractions of a second, up to six decimal places. Not all platforms support timestamps to a precision of six places.

Each symbol is substituted with the appropriate data for the time that is being formatted.

For symbols that represent numeric data, you can control zero-padding with the case of the symbols:

- Type the symbol in same-case (such as NN or nn) to allow zero padding. For example, hh:nn:ss.sss could produce 02:34:56.123.
- Type the symbol in mixed case (such as Nn) to suppress zero padding. For example, Hh:nn:ss.sss could produce 2:34:56.123.
- If the first two digits of the fractional seconds are mixed case (such as Ss or sSssss) then trailing zeros are removed. For example, hh:nn:ss.Sss could produce 12:34:56.1.

Remarks

Using mixed case in the formatting string suppresses leading zeros.

Related Information

sr_date_format Option [SQL Remote] [page 825]
sr_timestamp_format Option [SQL Remote] [page 828]
sr_timestamp_with_time_zone_format Option [SQL Remote] [page 830]
SQL Remote Options

1.5.6.7.128 sr_timestamp_format Option [SQL Remote]

Sets the format for TIMESTAMP values that are retrieved from the database.

Allowed Values

String

Default

yyyy/mm/dd hh:nn:ss.Ssssss

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

The Message Agent replicates DATETIME and TIMESTAMP information using this option.

The format is a string using the following symbols:

Symbol	Description
YY	Two-digit year.
YYYY	Four-digit year.
MM	Two-digit month, or two-digit minutes if following a colon (as in HH:MM).
MMM[m...]	Character short form for months (as many characters as there are "m"s).
DD	Two-digit day of month.
DDD[d...]	Character short form for day of the week.
HH	Two-digit hours.
NN	Two-digit minutes.
SS[.ssssss]	Seconds and fractions of a second, up to six decimal places. Not all platforms support timestamps to a precision of six places.
AA	A.M. or P.M. (12 hour clock). Omit AA and PP for 24 hour time.
PP	PM if needed (12 hour clock). Omit AA and PP for 24 hour time.

Each symbol is substituted with the appropriate data for the date that is being formatted.

For symbols that represent character data (such as MMM), you can control the case of the output as follows:

- Type the symbol in all uppercase to have the format appear in all uppercase. For example, MMM produces JAN.

- Type the symbol in all lowercase to have the format appear in all lowercase. For example, mmm produces jan.
- Type the symbol in mixed case to have the database server choose the appropriate case for the language that is being used. For example, in English, typing Mmm produces May, while in French it produces mai.

If the character data is multibyte, the length of each symbol reflects the number of characters, not the number of bytes. For example, the 'MMM' symbol specifies a length of three characters for the month.

For symbols that represent numeric data, you can control zero-padding with the case of the symbols:

- Type the symbol in same-case (such as MM or mm) to allow zero padding. For example, yyyy/mm/dd could produce 2002/01/01.
- Type the symbol in mixed case (such as Mm) to suppress zero padding. For example, yyyy/Mm/Dd could produce 2002/1/1.
- If the first two digits of the fractional seconds are mixed case (such as Ss or sSssss) then trailing zeros are removed. For example, hh:nn:ss.Sss could produce 12:34:56.1.

Related Information

sr_date_format Option [SQL Remote] [page 825]
sr_time_format Option [SQL Remote] [page 827]
sr_timestamp_with_time_zone_format Option [SQL Remote] [page 830]
SQL Remote Options

1.5.6.7.129 sr_timestamp_with_time_zone_format Option [SQL Remote]

Sets the format for TIMESTAMP WITH TIME ZONE values retrieved from the database.

Allowed Values

String

Default

yyyy/mm/dd hh:nn:ss.Ssssss +hh:nn

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

The Message Agent replicates TIMESTAMP WITH TIME ZONE information using this option.

The format is a string using the following symbols:

Symbol	Description
YY	Two-digit year.
YYYY	Four-digit year.
MM	Two-digit month, or two-digit minutes if following a colon (as in HH:MM).
MMM[m...]	Character short form for months (as many characters as there are "m"s).
DD	Two-digit day of month.
DDD[d...]	Character short form for day of the week.
HH	Two-digit hours.
NN	Two-digit minutes.
SS[.ssssss]	Seconds and fractions of a second, up to six decimal places. Not all platforms support timestamps to a precision of six places.
AA	A.M. or P.M. (12 hour clock). Omit AA and PP for 24 hour time.
PP	PM if needed (12 hour clock). Omit AA and PP for 24 hour time.

Each symbol is substituted with the appropriate data for the date that is being formatted.

For symbols that represent character data (such as MMM), you can control the case of the output as follows:

- Type the symbol in all uppercase to have the format appear in all uppercase. For example, MMM produces JAN.
- Type the symbol in all lowercase to have the format appear in all lowercase. For example, mmm produces jan.
- Type the symbol in mixed case to have the database server choose the appropriate case for the language that is being used. For example, in English, typing Mmm produces May, while in French it produces mai.

If the character data is multibyte, the length of each symbol reflects the number of characters, not the number of bytes. For example, the 'MMM' symbol specifies a length of three characters for the month.

For symbols that represent numeric data, you can control zero-padding with the case of the symbols:

- Type the symbol in same-case (such as MM or mm) to allow zero padding. For example, yyyy/mm/dd could produce 2002/01/01.
- Type the symbol in mixed case (such as Mm) to suppress zero padding. For example, yyyy/Mm/Dd could produce 2002/1/1.
- If the first two digits of the fractional seconds are mixed case (such as Ss or sSssss) then trailing zeros are removed. For example, hh:nn:ss.Sss could produce 12:34:56.1.

Related Information

sr_date_format Option [SQL Remote] [page 825]
sr_time_format Option [SQL Remote] [page 827]
sr_timestamp_format Option [SQL Remote] [page 828]
SQL Remote Options

1.5.6.7.130 st_geometry_asbinary_format Option

Controls how spatial values are converted from a geometry to binary.

Allowed Values

The list of values supported for this option is identical to the list of parameters (including sub-parameters) documented for the ST_AsBinary method.

Default

WKB

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION

	PUBLIC role	For current user	For other users
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option is used to determine how a geometry is converted to binary. The following are the two main contexts in which the option is used:

- fetching a value as binary from a client application
- executing a CAST statement. For example: `CAST( geometry-expression AS LONG BINARY )`

Assignments from geometry types to BINARY types give an error in contexts other than fetching from a client application and the CAST statement. For example, you cannot call a function or procedure that takes a BINARY parameter and pass a geometry, nor can you assign a geometry to a LONG BINARY variable.

Related Information

ST_AsBinary Method
st_geometry_astext_format Option [page 833]
st_geometry_asxml_format Option [page 834]
st_geometry_describe_type Option [page 836]
st_geometry_on_invalid Option [page 838]

1.5.6.7.131 st_geometry_astext_format Option

Controls how spatial values are converted from a geometry to text.

Allowed Values

The list of values supported for this option is identical to the list of parameters (including sub-parameters) documented for the ST_AsText method.

Default

WKT

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option is used to determine how a geometry is converted to text. The following are the two main contexts in which the option is used:

- fetching a value as text from a client application
- Executing a CAST statement. For example: `CAST( geometry-expression AS LONG VARCHAR )`

Assignments from geometry types to CHAR types give an error in contexts other than fetching from a client application and the CAST statement. For example, you cannot call a function or procedure that takes a CHAR parameter and pass a geometry, nor can you assign a geometry to a LONG VARCHAR variable.

Related Information

ST_AsText Method
st_geometry_asbinary_format Option [page 832]
st_geometry_asxml_format Option [page 834]
st_geometry_describe_type Option [page 836]
st_geometry_on_invalid Option [page 838]

1.5.6.7.132 st_geometry_asxml_format Option

Controls how spatial values are converted from a geometry to XML.

Allowed Values

The list of values supported for this option is identical to the list of parameters (including sub-parameters) documented for the ST_AsXML method.

Default

GML

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

This option is used to determine how a geometry is converted to XML. The following are the two main contexts in which the option is used:

- fetching a value as XML from a client application
- Executing a CAST statement. For example: `CAST( geometry-expression AS XML )`

Assignments from geometry types to XML give an error in contexts other than fetching from a client application and the CAST statement. For example, you cannot call a function or procedure that takes an XML parameter and pass a geometry, nor can you assign a geometry to an XML variable.

Related Information

ST_AsXML Method
st_geometry_asbinary_format Option [page 832]
st_geometry_astext_format Option [page 833]
st_geometry_describe_type Option [page 836]
st_geometry_on_invalid Option [page 838]

1.5.6.7.133 st_geometry_describe_type Option

Controls how spatial values are described.

Allowed Values

- CHAR
- NCHAR
- BINARY

Default

CHAR

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

Spatial values are fetched as character or binary because client libraries do not support spatial types directly. The st_geometry_describe_type specifies how a geometry column or expression is described to the client. Many client applications use the describe type for a column or expression to determine the client data type used to fetch the column or expression. Some client applications ignore the describe type, so setting the st_geometry_describe_type may not have the desired effect.

If a geometry is fetched as a character string type, the st_geometry_astext_format option is used to determine how to convert the geometry to text. If fetched as a binary type, the st_geometry_asbinary_format option is used. These are the same rules used by a CAST statement.

In ODBC applications, the SQL_DESC_TYPE_NAME and SQL_DESC_LOCAL_TYPE_NAME field identifiers of SQLColAttribute return the string `st_geometry` for columns of any geometry type. In JDBC, this is expressed by using the ResultSetMetaData.getColumnTypeName method.

Related Information

st_geometry_asbinary_format Option [page 832]
st_geometry_astext_format Option [page 833]
st_geometry_asxml_format Option [page 834]
st_geometry_on_invalid Option [page 838]

1.5.6.7.134 st_geometry_interpolation Option

Controls interpolation when spatial calculations involve circular arcs.

Allowed Values

This database option takes a semicolon-delimited set of key=value pairs (`key1=value1;key2=value2;...`). Allowed keys are relative-tolerance-percent and absolute-tolerance.

relative-tolerance-percent=DOUBLE

Use relative-tolerance-percent to specify the maximum variation, expressed as a percent, allowed in the distance from the center point of the ST_CircularString segment to the interpolated ST_LineString segment, relative to the radius of the ST_CircularString segment. The minimum allowed value for relative-tolerance-percent is 0, meaning the absolute-tolerance setting is always used.

absolute-tolerance=DOUBLE

Use absolute-tolerance to specify the maximum absolute variation, expressed in linear units, allowed in the distance from the center point of the ST_CircularString segment to the interpolated ST_LineString segment. The minimum value allowed for absolute-tolerance is also the tolerance specified by the SRID.

Defaults

If neither key is specified, relative-tolerance-percent defaults to 0.3, which gives 40 points in a complete circle, and absolute-tolerance defaults to the tolerance specified by the SRID.

If only one key is specified, the other key is set to its minimum value.

If both keys are specified, the minimum number of points is produced such that all restrictions are satisfied.

Default

relative-tolerance-percent=.3

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

Regardless of the setting for this database option, there are at least three points per curve segment. Furthermore, no two consecutive points on the interpolated ST_LineString are within tolerance of each other.

Additional points may be added to ensure the ST_Envelope of the ST_LineString is the same as that of the original ST_CircularString.

Related Information

How Interpolation Impacts Spatial Calculations
How Snap-to-grid and Tolerance Impact Spatial Calculations
ST_CircularString Type
ST_LineString Type
ST_Envelope Method

1.5.6.7.135 st_geometry_on_invalid Option

Controls the behavior when a geometry fails basic validation (for example, a linestring with one point, a polygon with a ring that is not closed, or a geometry that exceeds the bounds of the spatial reference system).

Allowed Values

Ignore

Do nothing when a geometry fails surface validation, and continue with the operation.

Error

Return an error when a geometry fails surface validation, and fail the operation.

Default

Error

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

If you must load invalid spatial data, you can set the option value to Ignore and then load the invalid data. Once loaded, you can use methods such as ST_IsSimple and ST_IsValue to determine the invalid row, ST_AsText to get the WKT representation of the data. Most other spatial methods give an error if they are passed invalid geometries. Use a text editor to correct the invalid geometry in the WKT, and then update the geometry in the database.

Related Information

st_geometry_asbinary_format Option [page 832]
st_geometry_astext_format Option [page 833]
st_geometry_asxml_format Option [page 834]
st_geometry_describe_type Option [page 836]

1.5.6.7.136 string_rtruncation Option

Determines whether an error is raised when a string is truncated.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

If the truncated characters or binary data consist only of spaces (x20), no exception is raised. The setting of On corresponds to ANSI/ISO SQL Standard behavior. When this option is set to Off, the exception is not raised and the character string is silently truncated.

String truncation may occur in several places. For example, using INSERT, UPDATE, CAST, or assignment to a variable may truncate a string if the declared destination type is too short. Truncation can occur for both character and binary data.

Related Information

Character Data Types
Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
SET Statement [T-SQL]

1.5.6.7.137 subscribe_by_remote Option [SQL Remote]

Controls the interpretation of NULL or empty-string SUBSCRIBE BY values.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

When the option is set to On, operations from remote databases on rows with a SUBSCRIBE BY value that is NULL or an empty string assume that the remote user is subscribed to the row. When it is set to Off, the remote user is assumed not to be subscribed to the row.

The only limitation of this option is that it leads to errors if a remote user really does want to INSERT (or UPDATE) a row with a NULL or empty subscription expression (for information held only at the consolidated database). This is reasonably obscure and can be worked around by assigning a subscription value in your installation that belongs to no remote user.

Related Information

SQL Remote Options
subscribe_by_remote Option with Many-to-many Relationships

1.5.6.7.138 subsume_row_locks Option

Controls when the database server acquires individual row locks for a table.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

If the subsume_row_locks option is On (the default) then whenever a table `t` is locked exclusively with LOCK TABLE `t` IN EXCLUSIVE MODE, the database server no longer acquires individual row locks for `t`.

This can result in a significant performance improvement if extensive updates are made to `t` in a single transaction, especially if `t` is large relative to cache size. It also allows for atomic update operations that are larger than the lock table can currently handle (approximately 2-4 million rows).

When this option is On, keyset cursors over a table locked in this fashion return row changed warnings for every row in the cursor, if any row in the database has been modified. The database server could turn an updatable cursor with an ORDER BY into a keyset cursor as a result.

Related Information

LOCK TABLE Statement

1.5.6.7.139 suppress_tds_debugging Option

Determines whether TDS debugging information appears in the database server messages window.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

When the database server is started with the -z option, debugging information appears in the database server messages window, including debugging information about the TDS protocol.

The suppress_tds_debugging option restricts the debugging information about TDS that appears in the database server messages window. When this option is set to Off (the default), TDS debugging information appears in the database server messages window.

Related Information

-z Database Server Option [page 532]

1.5.6.7.140 synchronize_mirror_on_commit Option

Controls when database changes are assured to have been sent to a mirror server when running in asynchronous or asyncfullpage mode.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The synchronize_mirror_on_commit option allows fine-grained control over when database changes are assured to have been sent to a mirror server when running in asynchronous or asyncfullpage mode. The option is Off by default. When set to On, each COMMIT causes any changes recorded in the transaction log to be sent to the mirror server, and an acknowledgement to be sent by the mirror server to the primary server once the changes are received by the mirror server.

The option can be set for specific transactions using SET TEMPORARY OPTION.

It may also be useful to set the option for specific applications by examining the APPINFO string in a login procedure. This allows mirroring behavior to be tailored to meet the needs of different applications.

Related Information

Database Mirroring [page 1757]

1.5.6.7.141 tds_empty_string_is_null Option

Controls whether empty strings are returned as NULL or a string containing one blank character for TDS connections.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

By default, this option is set to Off and empty strings are returned as a string containing one blank character for TDS connections. When this option is set to On, empty strings are returned as NULL strings for TDS connections. Non-TDS connections distinguish empty strings from NULL strings.

Related Information

NULL Special Value

1.5.6.7.142 temp_space_limit_check Option

Checks the amount of temporary file space used by a connection and fails the request if the amount of space requested is greater than the connection's allowable quota.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

When temp_space_limit_check is set to On (the default), if a connection requests more than its quota of temporary file space, then the request fails and the error SQLSTATE_TEMP_SPACE_LIMIT is returned. When this option is set to Off, the database server does not check the amount of temporary file space used by a connection. If a connection requests more than its quota of temporary space when this option is set to Off, a fatal error can occur.

The temporary file space quota for a connection is the minimum of the following two thresholds:

1. the maximum amount of temporary file space permitted for each connection as specified by the setting of the max_temp_space option
2. the maximum potential size of the temporary file divided by the number of connections

This threshold is used only if the temporary file has grown to 80% or more of its maximum size, which is determined by the amount of free space remaining on the device as reported by the operating system. When a connection requests more temporary file space than the quota allows, that connection's current request fails with SQLSTATE '54W05' (SQLSTATE_TEMP_SPACE_LIMIT).

You can specify a hard limit on the amount of temporary file space used by a connection with the max_temp_space option.

Related Information

Physical Limitations on Size and Number of Databases [page 936]
sa_disk_free_space System Procedure
max_temp_space Option [page 770]
-dt Database Server Option [page 416]

1.5.6.7.143 time_format Option

Sets the format for TIME values that are retrieved as strings from the database.

Allowed Values

String

Default

HH:NN:SS.SSS

A temporary setting for the current user is established by the ODBC and JDBC drivers ('hh:nn:ss'), and the OLE DB driver ('hh:nn:ss.ssssss').

A temporary setting for the current user is established by SAP Open Client and jConnect TDS connections ('HH:NN:SS.SSS').

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The format is a string using the following symbols:

Symbol	Description
HH	Two-digit hours.
NN	Two-digit minutes.
SS[.ssssss]	Seconds and fractions of a second, up to six decimal places. Not all platforms support timestamps to a precision of six places.
AA	AM or PM (12 hour clock). Omit AA and PP for 24 hour time.

Symbol	Description
PP	PM if needed (12 hour clock). Omit AA and PP for 24 hour time.

Each symbol is substituted with the appropriate data for the time that is being formatted.

For symbols that represent numeric data, you can control zero-padding with the case of the symbols:

- Type the symbol in same-case (such as NN or nn) to allow zero padding. For example, hh:nn:ss.sss could produce 02:34:56.123.
- Type the symbol in mixed case (such as Nn) to suppress zero padding. For example, Hh:nn:ss.sss could produce 2:34:56.123.
- If the first two digits of the fractional seconds are mixed case (such as Ss or sSssss) then trailing zeros are removed. For example, hh:nn:ss.Sss could produce 12:34:56.1.

Related Information

Open Client/jConnect TDS Compatibility Options [page 874]
Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
Retrieval of Dates and Times from the Database
Server Options Changed by ODBC
TIME Data Type
date_format Option [page 716]
timestamp_format Option [page 850]
timestamp_with_time_zone_format Option [page 852]
sp_tsql_environment System Procedure

1.5.6.7.144 time_zone_adjustment Option

Allows a connection's time zone adjustment to be modified.

Allowed Values

Unsigned integer (for example, 300)

A positive or negative signed integer enclosed in quotation marks (for example, '+300' or '-300')

String representing a time in hours and minutes, preceded by + or - and enclosed in quotation marks (for example, '+5:00', or '-5:00')

Default

0

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	No	No	No
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The time_zone_adjustment option value is the same value as that returned by `SELECT CONNECTION_PROPERTY( 'TimeZoneAdjustment' )`. The value represents the number of minutes that must be added to the Coordinated Universal Time (UTC) to display time local to the connection.

1.5.6.7.145 time_zone Option

Specifies which time zone the database uses for time zone calculations.

Allowed Values

String identifying an existing time zone that was created using the CREATE TIME ZONE statement.

Unsigned integer (for example, 300).

A positive or negative signed integer enclosed in quotation marks (for example, '+300' or '-300').

String representing a time in hours and minutes, preceded by + or - and enclosed in quotation marks (for example, '+5:00', or '-5:00').

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with MANAGE TIME ZONE	No	No
Allowed to set temporarily?	Yes, with MANAGE TIME ZONE	No	No

Remarks

Setting this option causes the database to act like it is running in the specified time zone. This database option affects calculations that use the CURRENT TIME special value or the NOW or GETDATE functions, but does not affect the value of the CURRENT UTC TIME or CURRENT SERVER TIME special value.

You can also set this option to an offset (number of minutes or hours:minutes) for the database to use an unnamed time zone with the specific offset and no daylight savings time.

> **i** Note
>
> To specify a string that identifies the time zone, first populate the ISYSTIMEZONE table with that time zone. To populate ISYSTIMEZONE, execute either the CREATE TIME ZONE or ALTER TIME ZONE statement.

1.5.6.7.146 timestamp_format Option

Sets the format for timestamps that are retrieved as strings from the database.

Allowed Values

String

Default

YYYY-MM-DD HH:NN:SS.SSS

A temporary setting for the current user is established by the ODBC, JDBC, and OLE DB drivers ('yyyy-mm-dd hh:nn:ss.ssssss').

A temporary setting for the current user is established by SAP Open Client and jConnect TDS connections ('YYYY-MM-DD HH:NN:SS.SSS').

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The format is a string using the following symbols:

Symbol	Description
YY	Two-digit year.
YYYY	Four-digit year.
MM	Two-digit month, or two-digit minutes if following a colon (as in HH:MM).
MMM[m...]	Character short form for months (as many characters as there are "m"s).
D	Single digit day of week (1 = Sunday, 7 = Saturday).
DD	Two-digit day of month.
DDD[d...]	Character short form for day of the week.
JJJ	Day of the year, from 1 to 366.
HH	Two-digit hours.
NN	Two-digit minutes.
SS[.ssssss]	Seconds and fractions of a second, up to six decimal places. Not all platforms support timestamps to a precision of six places.
AA	AM or PM (12 hour clock). Omit AA and PP for 24 hour time.
PP	PM if needed (12 hour clock). Omit AA and PP for 24 hour time.

Each symbol is substituted with the appropriate data for the date that is being formatted.

For symbols that represent character data (such as MMM), you can control the case of the output as follows:

- Type the symbol in all uppercase to have the format appear in all uppercase. For example, MMM produces JAN.
- Type the symbol in all lowercase to have the format appear in all lowercase. For example, mmm produces jan.
- Type the symbol in mixed case to have the database server choose the appropriate case for the language that is being used. For example, in English, typing Mmm produces May, while in French it produces mai.

If the character data is multibyte, the length of each symbol reflects the number of characters, not the number of bytes. For example, the 'mmm' symbol specifies a length of three characters for the month.

For symbols that represent numeric data, you can control zero-padding with the case of the symbols:

- Type the symbol in same-case (such as MM or mm) to allow zero padding. For example, yyyy/mm/dd could produce 2002/01/01.
- Type the symbol in mixed case (such as Mm) to suppress zero padding. For example, yyyy/Mm/Dd could produce 2002/1/1.
- If the first two digits of the fractional seconds are mixed case (such as Ss or sSssss) then trailing zeros are removed. For example, hh:nn:ss.Sss could produce 12:34:56.1.

> **i** Note
>
> If you change the setting for timestamp_format in a way that re-orders the date format, change the date_order option to reflect the same change, and vice versa.

Related Information

Open Client/jConnect TDS Compatibility Options [page 874]
Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
Server Options Changed by ODBC
TIMESTAMP Data Type
date_format Option [page 716]
date_order Option [page 719]
time_format Option [page 847]
timestamp_with_time_zone_format Option [page 852]
sp_tsql_environment System Procedure

1.5.6.7.147 timestamp_with_time_zone_format Option

Sets the format for TIMESTAMP WITH TIME ZONE values that are retrieved as strings from the database.

Allowed Values

String

Default

YYYY-MM-DD HH:NN:SS.SSS+HH:NN

A temporary setting for the current user is established by the ODBC, JDBC, and OLE DB drivers ('yyyy-mm-dd hh:nn:ss.ssssss +hh:nn').

A temporary setting for the current user is established by the .NET provider ('YYYY-MM-DD HH:NN:SS.SSSSSS +HH:NN').

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The format is a string using the following symbols:

Symbol	Description
YY	Two-digit year.
YYYY	Four-digit year.
MM	Two-digit month, or two-digit minutes if following a colon (as in HH:MM).
MMM[m...]	Character short form for months - as many characters as there are "m"s.
D	Single digit day of week (1 = Sunday, 7 = Saturday).
DD	Two-digit day of month.
DDD[d...]	Character short form for day of the week.
JJJ	Day of the year, from 1 to 366.
HH	Two-digit hours, or two digit hours of time zone offset if after '+'.
NN	Two-digit minutes, or two digit minutes of time zone offset if after '+'.
SS[.ssssss]	Seconds and fractions of a second, up to six decimal places. Not all platforms support timestamps to a precision of six places.
AA	AM or PM (12 hour clock). Omit AA and PP for 24 hour time.
PP	PM if needed (12 hour clock). Omit AA and PP for 24 hour time.

Symbol	Description
+	Indicates the start of a time zone offset. The '+' is replaced by a '+' or '-' depending on the time zone offset value.

Each symbol is substituted with the appropriate data for the date that is being formatted.

For symbols that represent character data (such as MMM), you can control the case of the output as follows:

- Type the symbol in all uppercase to have the format appear in all uppercase. For example, MMM produces JAN.
- Type the symbol in all lowercase to have the format appear in all lowercase. For example, mmm produces jan.
- Type the symbol in mixed case to have the database server choose the appropriate case for the language that is being used. For example, in English, typing Mmm produces May, while in French it produces mai.

If the character data is multibyte, the length of each symbol reflects the number of characters, not the number of bytes. For example, the mmm symbol specifies a length of three characters for the month.

For symbols that represent numeric data, you can control zero-padding with the case of the symbols:

- Type the symbol in same-case (such as MM or mm) to allow zero padding. For example, yyyy/mm/dd could produce 2002/01/01.
- Type the symbol in mixed case (such as Mm) to suppress zero padding. For example, yyyy/Mm/Dd could produce 2002/1/1.
- If the first two digits of the fractional seconds are mixed case (such as Ss or sSssss) then trailing zeros are removed. For example, hh:nn:ss.Sss could produce 12:34:56.1.

TIMESTAMP WITH TIME ZONE values can only be retrieved as strings from the database.

> **i** Note
>
> If you change the setting for timestamp_with_time_zone_format option in a way that re-orders the date format, then change the date_order option to reflect the same change, and vice versa.

Related Information

Server Options Changed by ODBC
TIMESTAMP WITH TIME ZONE Data Type
timestamp_format Option [page 850]
date_format Option [page 716]
date_order Option [page 719]
time_format Option [page 847]

1.5.6.7.148 truncate_timestamp_values Option

Limits the resolution of TIMESTAMP values.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

A TIMESTAMP value is precise to six decimal places in SQL Anywhere. However, to maintain compatibility with other software, which may truncate the TIMESTAMP value to three decimal places, you can set the truncate_timestamp_values option to On to limit the number of decimal places the database server stores. The default_timestamp_increment option determines the number of decimal places to which the TIMESTAMP value is truncated.

This option should not be enabled for databases already containing timestamp data.

For MobiLink synchronization, if you are going to set this option, it must be set before performing the first synchronization.

If the database server finds TIMESTAMP values with a higher resolution than that specified by the combination of truncate_timestamp_values and default_timestamp_increment, an error is reported.

Unloading the database and then reloading it into a new database in which the truncate_timestamp_values and default_timestamp_increment values have been set is usually the easiest solution to ensure that the proper

TIMESTAMP values are used. However, depending on the type of TIMESTAMP columns in your table, you can also do the following:

- If the TIMESTAMP columns are defined with DEFAULT TIMESTAMP or DEFAULT UTC TIMESTAMP (so that the value is automatically updated by the database server when the row is modified), you must delete all the rows in the table before the truncate_timestamp_values option is changed. You can delete the rows using the DELETE or TRUNCATE TABLE statement.
- If the TIMESTAMP column is defined with a value other than DEFAULT TIMESTAMP or DEFAULT UTC TIMESTAMP, you can execute an UPDATE statement that casts the values to a string and then back to a TIMESTAMP. For example:

  ```
  UPDATE T
     SET ts = CAST( DATEFORMAT( ts, 'yyyy/mm/dd hh:nn:ss.ss' )
   AS TIMESTAMP );
  ```

  This process may lose more precision than is necessary. The format string to use depends on the number of digits of precision to be kept.

Example

Setting the default_timestamp_increment option to 100000 causes truncation after the first decimal place in the seconds component, allowing a value such as '2000/12/05 10:50:53:700' to be stored.

Related Information

default_timestamp_increment Option [page 725]
TIMESTAMP Special Value
CURRENT UTC TIMESTAMP Special Value
UTC TIMESTAMP Special Value

1.5.6.7.149 trusted_certificates_file Option

Specifies the file that contains the list of trusted Certificate Authority certificates when the database server acts as a client to an LDAP server.

Allowed Values

String

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SECURITY OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SECURITY OPTION	No	No

Remarks

This option does not apply when the database server is a client for a secure web procedure.

An incoming TLS connection to the database server uses certificate settings set by the -ec server option. For the database server to act as a client to another server (for example, when the database server connects to an LDAP server) the Certificate Authority (CA) that signed the TLS certificate must be known.

For client connections, the trusted_certificates connection option is set with the path to a file containing a list of trusted CA certificates. Similarly, the trusted_certificates_file database option specifies trusted CA certificates when the database server acts as a client.

Setting this option to `*` or leaving it empty causes the software to use a certificate from the operating system certificate store if the LDAP URL begins with `ldaps://` or if the LDAP server has been configured to use the TLS protocol.

When this option is set to NULL, no outbound TLS connections can be started because there are no trusted Certificate Authorities.

Example

In this example, the list of trusted certificate authorities that sign server certificates is found in a local file called `trusted.txt`:

```
SET OPTION PUBLIC.trusted_certificates_file = 'C:\\certificates\\shared\
\trusted.crt';
```

In the following two examples, the list of trusted certificate authorities that sign server certificates is found in the operating system certificate store:

```
SET OPTION PUBLIC.trusted_certificates_file = '*';
```

```
SET OPTION PUBLIC.trusted_certificates_file = '';
```

Related Information

-ec Database Server Option [page 418]
trusted_certificate Protocol Option [page 244]

1.5.6.7.150 tsql_outer_joins Option

Controls the ability to use the Transact-SQL outer join operators *= and =* in statements and views.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

Support for Transact-SQL outer joins is deprecated. Setting this option to On allows you to use Transact-SQL outer joins.

Related Information

Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
Transact-SQL Outer Joins (*= or =*)
Join Operators

1.5.6.7.151 tsql_variables Option

Controls whether the @ sign can be used as a prefix for Embedded SQL host variable names.

Allowed Values

On, Off

Default

Off

A temporary setting for the current user is established by SAP Open Client and jConnect TDS connections ('On').

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

When this option is set to On, you can use the @ sign instead of the colon as a prefix for host variable names in Embedded SQL. This is implemented primarily for Transact-SQL compatibility.

Related Information

Open Client/jConnect TDS Compatibility Options [page 874]
Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
sp_tsql_environment System Procedure

1.5.6.7.152 updatable_statement_isolation Option

Specifies the isolation level used by updatable statements when the isolation_level option is set to Readonly-statement-snapshot.

Allowed Values

0, 1, 2, 3

Default

0

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The isolation level specified by the updatable_statement_isolation option is used by updatable statements when the isolation_level option is set to Readonly-statement-snapshot. The following values are accepted:

0

Allow dirty reads, non-repeatable reads, and phantom rows.

1

Prevent dirty reads. Allow non-repeatable reads and phantom rows.

2

Prevent dirty reads and non-repeatable reads. Allow phantom rows.

3

Serializable. Prevent dirty reads, non-repeatable reads, and phantom rows.

Related Information

Snapshot Isolation
Isolation Levels and Consistency
Guidelines for Choosing Isolation Levels
Typical Types of Inconsistency
isolation_level Option [page 746]

1.5.6.7.153 update_statistics Option

Controls whether connections can send query feedback to the statistics governor.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

The database server collects statistics during normal query execution and uses the gathered statistics to self-tune the column statistics. When you set this option to Off, the statistics governor does not receive feedback about the health of statistics or fix any statistics for the specified connection. However, it does continue to monitor the usage of statistics and create or drop statistics based on their usage.

When this option is set by a user with the SET ANY PUBLIC OPTION system privilege, its setting affects all connections to the database; otherwise, only the current users connections are affected.

Under normal circumstances, it should not be necessary to turn this option off.

The update_statistics option does not affect changes to statistics as a result of updates to data (LOAD/INSERT/UPDATE/DELETE). To control whether statistics are updated based on these statements, use the collect_statistics_on_dml_updates database option.

The difference between the collect_statistics_on_dml_updates option and the update_statistics option is that the update_statistics option compares the actual number of rows that satisfy a predicate with the number of rows that are estimated to satisfy the predicate, and then updates the estimates. The collect_statistics_on_dml_updates option modifies the column statistics based on the values of the specific rows that are inserted, updated, or deleted.

Related Information

How the Statistics Governor Maintains Statistics [page 1449]
collect_statistics_on_dml_updates Option [page 704]

1.5.6.7.154 user_estimates Option

Controls whether user selectivity estimates in query predicates are respected or ignored by the query optimizer.

Allowed Values

Enabled, Disabled, Override-Magic

Default

Override-magic

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

You can specify user selectivity estimates to improve the optimizer's performance when the database server is unable to accurately predict the selectivity of a predicate. However, user selectivity estimates should be used only in appropriate circumstances. For example, it may be useful to supply a selectivity estimate for a predicate that involves one or more functions if the Override-Magic selectivity estimate used by the optimizer is significantly different from the actual selectivity.

If you have used selectivity estimates that are inaccurate as a workaround to performance problems where the software-selected access plan was poor, set this option to Disabled. The database server may not select an optimal plan if you use inaccurate estimates.

When a user selectivity estimate is supplied with a predicate, the estimate is respected or ignored based on the setting of this option. The following values are accepted:

Enabled

All user-supplied selectivity estimates are respected. You can also use On to turn on this option.

Override-Magic

A user selectivity estimate is respected and used only if the optimizer would otherwise choose to use its last-resort, heuristic value (also called the magic value).

Disabled

All user estimates are ignored and magic values are used when no other estimate data is available. You can also use Off to turn off this option.

You can override any temporary or PUBLIC settings for this option within individual INSERT, UPDATE, DELETE, SELECT, UNION, EXCEPT, and INTERSECT statements by including an OPTION clause in the statement.

Related Information

Explicit Selectivity Estimates
Explicit Selectivity Estimates
INSERT Statement
UPDATE Statement
DELETE Statement
SELECT Statement
UNION Statement

EXCEPT Statement
INTERSECT Statement

1.5.6.7.155 uuid_has_hyphens Option

Controls the formatting of unique identifier values when they are converted to strings.

Allowed Values

On, Off

Default

On

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

When the uuid_has_hyphens option is set to On, the resulting strings contain four hyphens. For example, 12345678-1234-1234-1234-1234567890ab. With this option set to Off, the hyphens are omitted from the string. The database server supports UUID strings both with and without hyphens when converting a string to a unique identifier value.

Related Information

UNIQUEIDENTIFIER Data Type

1.5.6.7.156 verify_all_columns Option [SQL Remote]

Controls whether messages that contain updates published by the local database are sent with all column values included.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

This option is used by SQL Remote only. When it is set to On, messages that contain updates published by the local database are sent with all column values included, and a conflict in any column triggers a RESOLVE UPDATE trigger at the subscriber database.

Example

The following statement sets the verify_all_columns option to Off in SQL Anywhere, for all users:

```
SET OPTION PUBLIC.verify_all_columns = 'Off';
```

Related Information

SQL Remote Options
The verify_all_columns Option

1.5.6.7.157 verify_password_function Option

Implements password rules.

Allowed Values

String

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SECURITY OPTION	Yes, with SET ANY SECURITY OPTION	Yes, with SET ANY SECURITY OPTION
Allowed to set temporarily?	Yes, with SET ANY SECURITY OPTION	Yes (current connection only), with SET ANY SECURITY OPTION	No

Remarks

If a password is set, no function is called if the value is an empty string.

The function specified by this option is called automatically when a non-NULL password is created or set. The function specified is not called for users that have a login policy with the change_password_dual_control option enabled. To prevent a user from overriding the function, set the option value to `owner.function-name`. A user must have a password to connect to the database.

After validating the statement used to create or set the password, the function is called to verify the password using the specified rules. If the password conforms to the specified rules, the function returns NULL to indicate

success, and the invoking statement executes. Otherwise a non-NULL string is returned, and is cited as the reason for the error.

The password verification function takes two parameters: `user-name` VARCHAR(128) and `new-pwd` VARCHAR(255). It returns a value of type VARCHAR(255).

Execute an ALTER FUNCTION `function-name` SET HIDDEN statement on the password verification function to ensure that it cannot be stepped through using the debugger.

If the verify_password_function option is set, specifying more than one user ID and password with a GRANT CONNECT statement is not allowed.

Example

The following example defines a table and a function and sets some login policy options. Together they implement advanced password rules that include requiring certain types of characters in the password, disallowing password re-use, and expiring passwords. The function is called by the database server with the verify_password_function option when a user ID is created or a password is changed. The application can call the procedure specified by the post_login_procedure option to report that the password should be changed before it expires.

The code for this sample is also available in the following location: *%SQLANYSAMP17%*`\SQLAnywhere\SQL\verify_password.sql`.

```
-- Only a database administrator should have permissions on this table.
CREATE TABLE IF NOT EXISTS DBA.t_pwd_history(
        pk          INT         DEFAULT AUTOINCREMENT PRIMARY KEY,
        user_name   CHAR(128),          -- the user whose password is set
        pwd_hash    CHAR(32) );         -- hash of password value to detect
                                        -- duplicate passwords
-- Called whenever a non-NULL password is set to verify that the password
-- conforms to password rules.
CREATE OR REPLACE FUNCTION DBA.f_verify_pwd( uid     VARCHAR(128),
                                             new_pwd VARCHAR(255) )
RETURNS VARCHAR(255)
BEGIN
    -- a table with one row per character in new_pwd
    DECLARE local temporary table pwd_chars(
            pos INT PRIMARY KEY,    -- index of c in new_pwd
            c   CHAR( 1 CHAR ) );   -- character
    -- new_pwd with non-alpha characters removed
    DECLARE pwd_alpha_only      CHAR(255);
    DECLARE num_lower_chars     INT;
    -- enforce minimum length (can also be done with
    -- min_password_length option)
    IF length( new_pwd ) < 6 THEN
        RETURN 'password must be at least 6 characters long';
    END IF;
    -- break new_pwd into one row per character
    INSERT INTO pwd_chars SELECT row_num, substr( new_pwd, row_num, 1 )
                            FROM dbo.RowGenerator
                            WHERE row_num <= length( new_pwd );
    -- copy of new_pwd containing alpha-only characters
    SELECT list( c, '' ORDER BY pos ) INTO pwd_alpha_only
        FROM pwd_chars WHERE c BETWEEN 'a' AND 'z' OR c BETWEEN 'A' AND 'Z';
    -- number of lower case characters IN new_pwd
    SELECT count(*) INTO num_lower_chars
        FROM pwd_chars WHERE CAST( c AS BINARY ) BETWEEN 'a' AND 'z';
    -- enforce rules based on characters contained in new_pwd
```

```
    IF ( SELECT count(*) FROM pwd_chars WHERE c BETWEEN '0' AND '9' )
           < 1 THEN
        RETURN 'password must contain at least one numeric digit';
    ELSEIF length( pwd_alpha_only ) < 2 THEN
        RETURN 'password must contain at least two letters';
    ELSEIF num_lower_chars = 0
           OR length( pwd_alpha_only ) - num_lower_chars = 0 THEN
        RETURN 'password must contain both upper- and lowercase characters';
    END IF;
    -- not the same as any user name
    -- (this could be modified to check against a disallowed words table)
    IF EXISTS( SELECT * FROM SYS.SYSUSER
                   WHERE lower( user_name ) IN ( lower( pwd_alpha_only ),
                                                 lower( new_pwd ) ) ) THEN
        RETURN 'password or only alphabetic characters in password ' ||
               'must not match any user name';
    END IF;
    -- not the same as any previous password for this user
    IF EXISTS( SELECT * FROM t_pwd_history
                   WHERE user_name = uid
                     AND pwd_hash = hash( uid || new_pwd, 'md5' ) ) THEN
        RETURN 'previous passwords cannot be reused';
    END IF;
    -- save the new password
    INSERT INTO t_pwd_history( user_name, pwd_hash )
        VALUES( uid, hash( uid || new_pwd, 'md5' ) );
    RETURN( NULL );
END;
ALTER FUNCTION DBA.f_verify_pwd SET HIDDEN;
GRANT EXECUTE ON DBA.f_verify_pwd TO PUBLIC;
SET OPTION PUBLIC.verify_password_function = 'DBA.f_verify_pwd';
-- All passwords expire in 180 days.  Expired passwords can be changed
-- by the user using the NewPassword connection parameter.
ALTER LOGIN POLICY root password_life_time = 180;
-- If an application calls the procedure specified by the post_login_procedure
-- option, then the procedure can be used to warn the user that their
-- password is about to expire.
ALTER LOGIN POLICY root password_grace_time = 30;
-- Five consecutive failed login attempts will result in a userid being locked.
ALTER LOGIN POLICY root max_failed_login_attempts = 5;
```

Related Information

Password and User ID Restrictions and Considerations [page 1634]
Security: Passwords [page 1685]
Security: Passwords [page 1685]
Dual Control Passwords [page 1639]
min_password_length Option [page 772]
CREATE USER Statement
CREATE FUNCTION Statement
ALTER FUNCTION Statement
ALTER USER Statement
NewPassword (NEWPWD) Connection Parameter [page 98]

1.5.6.7.158 verify_threshold Option [SQL Remote]

Controls which columns are verified when updates are replicated.

Allowed Values

Integer, in bytes

Default

1000

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY USER DEFINED OPTION	Yes	Yes, with SET ANY USER DEFINED OPTION
Allowed to set temporarily?	No	No	No

Remarks

This option is used by SQL Remote only. If the data type of a column is longer than the threshold, old values for the column are not verified when an UPDATE is replicated. This keeps the size of SQL Remote messages down, but has the disadvantage that conflicting updates of long values are not detected.

Related Information

SQL Remote Options
BLOBs

1.5.6.7.159 wait_for_commit Option

Determines when foreign key integrity is checked, as data is manipulated.

Allowed Values

On, Off

Default

Off

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY PUBLIC OPTION	Yes	Yes, with SET ANY PUBLIC OPTION
Allowed to set temporarily?	Yes, with SET ANY PUBLIC OPTION	Yes (current connection only)	No

Remarks

If this option is set to On, the database does not check foreign key integrity until the next COMMIT statement. Otherwise, all foreign keys that are not created with the CHECK ON COMMIT clause are checked as they are inserted, updated, or deleted.

Related Information

Locks During Inserts
Referential Integrity Checking
sa_check_commit System Procedure
COMMIT Statement

1.5.6.7.160 webservice_namespace_host Option

Specifies the hostname within the XML namespace specification for DISH services.

Allowed Values

NULL or hostname-string (which can be empty)

Default

Empty string

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

This option overrides the default usage of the HTTP Host request header.

DISH services export Webservices Description Language Documents (WSDLs). These are XML documents that contain descriptions of the available SOAP services. The URL of the targetNameSpace and the soapAction operations within this XML document contain a hostname. When this option is set to NULL, the default value, the hostname is that of the computer on which the database server is running. If this option is set to a string value, the string is used as the hostname instead. This option is intended for use when developing web service client applications that, when deployed, target a host other than the one used for development.

Related Information

The Database Server as an HTTP Web Server

1.5.6.7.161 webservice_sessionid_name Option

Redefines what the web server uses to determine whether session management is used.

Allowed Values

NULL or sessionid-string

Default

SessionID

Scope

	PUBLIC role	For current user	For other users
Allowed to set permanently?	Yes, with SET ANY SYSTEM OPTION	No	No
Allowed to set temporarily?	Yes, with SET ANY SYSTEM OPTION	No	No

Remarks

The webservice_sessionid_name option allows you to redefine what the web server looks for to determine whether session management is being used. By default, the session identifier name is SessionID.

This database option is useful if your web application uses a name other than SessionID for the session identifier name, for example JSessionID.

When the web application uses URL-based session management, then the session identifier appears in the GET request:

```
GET /service?JSessionID=session_63520025849921 HTTP/1.1
```

When the web application uses cookie-based session management, the session identifier appears in the Cookie header.

```
Cookie: JSessionID=cookie_session_63520025481454
```

The name specified by the webservice_sessionid_name database option is only used when the web server analyzes web service requests.

Example

- The following example renames the web service SessionID to JSessionID:

```
SET OPTION PUBLIC.webservice_sessionid_name="JSessionID";
```

Related Information

The Database Server as an HTTP Web Server

1.5.6.8 Different Categories of Database Options

There are four categories of database options: compatibility, synchronization, SQL Remote, and Interactive SQL.

In this section:

Open Client/jConnect TDS Compatibility Options [page 874]
When a connection is made to a database using SAP Open Client or jConnect, some option settings are explicitly set for the current connection to be compatible with Adaptive Server Enterprise.

Transact-SQL and ANSI/ISO SQL Standard Compatibility Options [page 876]
There are several compatibility options that allow you to make SQL Anywhere behavior compatible with Adaptive Server Enterprise, or to support both old behavior and allow ANSI/ISO SQL Standard behavior.

Synchronization Options [page 877]
There are several database options that can be set to configure SQL Anywhere databases used as MobiLink synchronization clients:

SQL Remote Options [page 878]
There are several options that control SQL Remote replication behavior.

Related Information

Interactive SQL Options [page 1054]

1.5.6.8.1 Open Client/jConnect TDS Compatibility Options

When a connection is made to a database using SAP Open Client or jConnect, some option settings are explicitly set for the current connection to be compatible with Adaptive Server Enterprise.

The default database options are set for TDS connections using a system procedure named sp_tsql_environment.

> **i** Note
>
> Do not alter the sp_tsql_environment procedure. It is for system use only.

Although the database server allows long user names and passwords, TDS client names and passwords cannot exceed 30 bytes. If your password or user ID is longer than 30 bytes, attempts to connect over TDS (for example, using jConnect) return an invalid user or password error.

Option	Setting
allow_nulls_by_default option	Off
ansi_blanks option	On
ansinull option	Off
chained option	Off
close_on_endtrans option	Off (for SQL Anywhere databases only)
date_format option	YYYY-MM-DD
date_order option	MDY
escape_character option (reserved for system use)	Off
isolation_level option	1
on_tsql_error option	Continue
quoted_identifier option	Off
time_format option	HH:NN:SS.SSS
timestamp_format option	YYYY-MM-DD HH:NN:SS.SSS
tsql_variables option	On

In this section:

Changing the Option Settings for a TDS Connection [page 875]
Change the options for a TDS connection.

Related Information

allow_nulls_by_default Option [page 672]
ansi_blanks Option [page 677]
ansinull Option [page 684]

chained Option [page 699]
close_on_endtrans Option [page 702]
date_format Option [page 716]
date_order Option [page 719]
escape_character Option (Reserved for System Use) [page 732]
isolation_level Option [page 746]
on_tsql_error Option [page 784]
quoted_identifier Option [page 804]
time_format Option [page 847]
timestamp_format Option [page 850]
tsql_variables Option [page 859]

1.5.6.8.1.1 Changing the Option Settings for a TDS Connection

Change the options for a TDS connection.

Prerequisites

You must have the CREATE PROCEDURE system privilege to create procedures owned by you.

You must have the SET ANY SECURITY OPTION system privilege to set the login_procedure option.

Context

When the database server is serving applications over TDS, it automatically sets relevant database options to values compatible with Adaptive Server Enterprise default behavior.

Procedure

1. Create a procedure that sets the database options you want.
2. Set the PUBLIC login_procedure option to the name of the new procedure.

Results

Future connections use the procedure. You can also configure a different procedure for different user IDs.

Example

This procedure example changes only the quoted_identifier option from the default setting.

```
CREATE PROCEDURE DBA.my_startup_procedure()
BEGIN
  CALL dbo.sp_login_environment();
  IF CONNECTION_PROPERTY('CommProtocol')='TDS' THEN
    SET TEMPORARY OPTION quoted_identifier='On';
  END IF
END;
GRANT EXECUTE ON DBA.my_startup_procedure TO PUBLIC;
```

Run the following command to set the PUBLIC login_procedure option to the name of the new procedure:

```
SET OPTION PUBLIC.login_procedure = 'DBA.my_startup_procedure';
```

Run the following command to set the login_procedure option to the name of the new procedure for the user Browser only:

```
SET OPTION Browser.login_procedure = 'DBA.my_startup_procedure';
```

Related Information

Database Options [page 651]
login_procedure Option [page 755]
CREATE PROCEDURE Statement

1.5.6.8.2 Transact-SQL and ANSI/ISO SQL Standard Compatibility Options

There are several compatibility options that allow you to make SQL Anywhere behavior compatible with Adaptive Server Enterprise, or to support both old behavior and allow ANSI/ISO SQL Standard behavior.

Where SAP Adaptive Server Enterprise has a comparable option, these entries are flagged ASE.

The default setting for some of these options differs from the Adaptive Server Enterprise default setting. To ensure compatibility across your SQL Anywhere and Adaptive Server Enterprise databases, explicitly set each of the compatibility options.

For further compatibility with Adaptive Server Enterprise, some of these options can be set for the duration of the current connection using the Transact-SQL SET statement instead of the SQL Anywhere SET OPTION statement.

- allow_nulls_by_default option
- ansi_blanks option
- ansi_close_cursors_on_rollback option

- ansi_permissions option (ASE)
- ansi_substring option
- ansi_update_constraints option
- ansinull option (ASE)
- chained option (ASE)
- close_on_endtrans option (ASE)
- continue_after_raiserror option
- conversion_error option
- date_format option (ASE)
- date_order option
- divide_by_zero_error option
- escape_character option (reserved for system use)
- fire_triggers option
- first_day_of_week option (ASE)
- isolation_level option (ASE)
- nearest_century option
- non_keywords option
- on_tsql_error option
- quoted_identifier option (ASE)
- reserved_keywords option
- string_rtruncation option (ASE)
- time_format option
- timestamp_format option
- tsql_outer_joins option
- tsql_variables option
- SET statement [T-SQL]

1.5.6.8.3 Synchronization Options

There are several database options that can be set to configure SQL Anywhere databases used as MobiLink synchronization clients:

Option	Values	Default
default_timestamp_increment option	Integer, representing microseconds, from 1 to 1000000	1
delete_old_logs option [SQL Remote]	On, Off, Delay, `n` days	Off
ml_remote_id option [MobiLink]	Any value that uniquely identifies the database for MobiLink synchronization	NULL
prevent_article_pkey_update option	On, Off	On
truncate_timestamp_values option	On, Off	Off

Related Information

default_timestamp_increment Option [page 725]
delete_old_logs Option [MobiLink][SQL Remote] [page 729]
ml_remote_id Option [MobiLink] [page 774]
prevent_article_pkey_update Option [MobiLink] [page 797]
truncate_timestamp_values Option [page 855]

1.5.6.8.4 SQL Remote Options

There are several options that control SQL Remote replication behavior.

Option	Values	Default
blob_threshold option [SQL Remote]	Integer (in bytes)	256
compression option [SQL Remote]	Integer, from -1 to 9	6
delete_old_logs option [SQL Remote]	On, Off, Delay, `n` days	Off
external_remote_options option [SQL Remote]	On, Off	Off
qualify_owners option [SQL Remote]	On, Off	On
quote_all_identifiers option [SQL Remote]	On, Off	Off
replication_error option [SQL Remote]	Stored procedure name	(no procedure)
replication_error_piece option [SQL Remote]	Stored procedure name	(no procedure)
save_remote_passwords option [SQL Remote]	On, Off	On
sr_date_format option [SQL Remote]	`date-string`	yyyy/mm/dd
sr_time_format option [SQL Remote]	`time-string`	hh:nn:ss.Ssssss
sr_timestamp_format option [SQL Remote]	`timestamp-string`	yyyy/mm/dd hh:nn:ss.Ssssss
sr_timestamp_with_time_zone_format option [SQL Remote]	`timestamp-with-time-zone-string`	yyyy/mm/dd hh:nn:ss.Ssssss +hh:nn
subscribe_by_remote option [SQL Remote]	On, Off	On
verify_all_columns option [SQL Remote]	On, Off	Off
verify_threshold option [SQL Remote]	Integer (in bytes)	1000

Related Information

blob_threshold Option [SQL Remote] [page 693]
compression Option [SQL Remote] [page 705]
delete_old_logs Option [MobiLink][SQL Remote] [page 729]
external_remote_options Option [SQL Remote] [page 735]
qualify_owners Option [SQL Remote] [page 801]
quote_all_identifiers Option [SQL Remote] [page 803]
replication_error Option [SQL Remote] [page 808]
replication_error_piece Option [SQL Remote] [page 809]
save_remote_passwords Option [SQL Remote] [page 817]
sr_date_format Option [SQL Remote] [page 825]
sr_time_format Option [SQL Remote] [page 827]
sr_timestamp_format Option [SQL Remote] [page 828]
sr_timestamp_with_time_zone_format Option [SQL Remote] [page 830]
subscribe_by_remote Option [SQL Remote] [page 840]
verify_all_columns Option [SQL Remote] [page 865]
verify_threshold Option [SQL Remote] [page 869]

1.5.7 Connection, Database, and Database Server Properties

Connection, database, and database server properties control how data is accessed and processed in the database.

In this section:

List of Connection Properties [page 880]
Connection properties are available for each connection to a database. Connection property names are case insensitive. Use the CONNECTION_PROPERTY system function or the sa_conn_properties system procedure to retrieve connection properties.

List of Database Server Properties [page 900]
Database server properties are available for each connection to a database. Use the PROPERTY system function to retrieve the value for an individual property and use the sa_eng_properties system procedure to retrieve the values of all database server properties.

List of Database Properties [page 917]
Database properties are available for each connection to a database. Use the DB_PROPERTY system function to retrieve the value for an individual database property, or the sa_db_properties system procedure to retrieve the values of all database properties.

User-defined Properties [page 932]
User-defined properties are available for each connection to a database.

Database Server Property Tracking [page 933]
The database server can store the values of numeric database server properties so that you can track the changes of numeric database server properties over time.

1.5.7.1 List of Connection Properties

Connection properties are available for each connection to a database. Connection property names are case insensitive. Use the CONNECTION_PROPERTY system function or the sa_conn_properties system procedure to retrieve connection properties.

Example

The following statement returns the number of pages that have been read from file by the current connection.

```
SELECT CONNECTION_PROPERTY ( 'DiskRead' );
```

Use the sa_conn_properties system procedure to retrieve the values of all connection properties:

```
CALL sa_conn_properties( );
```

Connection Properties

Property name	Description
allow_nulls_by_default	Whether columns created without specifying either NULL or NOT NULL are allowed to contain NULL values. This property corresponds to the allow_nulls_by_default option for the connection.
allow_read_client_file	Whether the database server allows the reading of files on a client computer. This property corresponds to the allow_read_client_file option for the connection.
allow_snapshot_isolation	Whether snapshot isolation is enabled or disabled. This property corresponds to the allow_snapshot_isolation option for the connection.
allow_write_client_file	Whether the database server allows the writing of files to a client computer. This property corresponds to the allow_write_client_file option for the connection.
ansi_blanks	Indicates when character data is truncated at the client side. This property corresponds to ansi_blanks option for the connection.
ansi_close_cursors_on_rollback	Whether cursors opened WITH HOLD are closed when a ROLLBACK is performed. This property corresponds to the ansi_close_cursors_on_rollback option for the connection.
ansi_permissions	Whether privileges are checked for DELETE and UPDATE statements. This property corresponds to the ansi_permissions option for the connection.
ansi_substring	The behavior of the SUBSTRING (SUBSTR) function when negative values are provided for the start or length parameters. This property corresponds to the ansi_substring option for the connection.

Property name	Description
ansi_update_constraints	The range of updates that are permitted. This property corresponds to the ansi_update_constraints option for the connection.
ansinull	How NULL values are interpreted. This property corresponds to the ansinull option.
AppInfo	Information about the client that made the connection. For HTTP connections, this includes information about the browser. For connections using older versions of SAP Open Client or jConnect, the information may be incomplete.
ApproximateCPUTime	The estimate of the amount of CPU time accumulated by a given connection, in seconds. The value returned may differ from the actual value by as much as 50%, although typical variations are in the 5-10% range. On multi-processor computers, each CPU (or hyperthread or core) accumulates time, so the sum of accumulated times for all connections may be greater than the elapsed time.
audit_log	The location where audit logs are directed.
auditing	Whether auditing is enabled for the database(On) or not (Off). This option corresponds to the auditing option.
auditing_options	This property is reserved for system use.
Authenticated	Whether the application sent a valid connection authentication string (Yes) or not (No).
AuthType	The type of authentication used when connecting. The value returned is one of Standard, Integrated, Kerberos, LDAPUA, CloudAdmin,or an empty string. The value is an empty string when the connection is an internal connection or for connections for HTTP services that use AUTHORIZATION OFF.
auto_commit	Whether the database server automatically commits after each statement. By default, the database server operates in manual commit mode. To turn on automatic commits, set the auto_commit database option (a server-side option). Do not confuse this option with the Interactive SQL option of the same name.
auto_commit_on_create_local_temp_index	Whether the database server performs a COMMIT before an index is created on a local temporary table. This property corresponds to the value of the auto_commit_on_create_local_temp_index option.
background_priority	This property is deprecated. The value of the background_priority option for the connection, which indicates how much impact the current connection has on the performance of other connections.
BlockedOn	Whether the current connection is blocked or not (zero). When the connection is blocked because of a locking conflict, the value is the connection number on which the connection is blocked.
blocking	The database server's behavior in response to locking conflicts. This property corresponds to the blocking option for the connection.

Property name	Description
blocking_others_timeout	The length of time that another connection can block on the current connection's row and table locks before the current connection is rolled back. This property corresponds to the value of the blocking_others_timeout option.
blocking_timeout	The length of time, in milliseconds, a transaction waits to obtain a lock. This property corresponds to the blocking_timeout option.
BytesReceived	The number of bytes received during client/server communications. This value is updated for HTTP and HTTPS connections.
BytesReceivedUncomp	The number of bytes that would have been received during client/server communications if compression was disabled. This value is the same as the value for BytesReceived if compression is disabled.
BytesSent	The number of bytes sent during client/server communications. This value is updated for HTTP and HTTPS connections.
BytesSentUncomp	The number of bytes that would have been sent during client/server communications if compression was disabled. This value is the same as the value for BytesSent if compression is disabled.
CacheHits	The number of successful reads of the cache.
CacheRead	The number of database pages that have been looked up in the cache.
CacheReadIndInt	The number of index internal-node pages that have been read from the cache.
CacheReadIndLeaf	The number of index leaf pages that have been read from the cache.
CacheReadTable	The number of table pages that have been read from the cache.
CacheReadWorkTable	The number of cache work table reads.
CarverHeapPages	The number of heap pages used for short-term purposes such as query optimization.
chained	The value of the chained option, which indicates the transaction mode used in the absence of a BEGIN TRANSACTION statement.
CharSet	The CHAR character set used by the connection. This property has extensions that you can specify when querying the property value.
checkpoint_time	The value of the checkpoint_time option, which indicates the maximum time, in minutes, that the database server runs without doing a checkpoint.
cis_option	Specifies 7 if debugging information for remote data access appears in the database server messages window. Specifies 0 if the debugging information for remote data access does not appear in the database server messages window. This property corresponds to the cis_option option.
cis_rowset_size	The number of rows that are returned from remote servers for each fetch. This property corresponds to the value of the cis_rowset_size option.

Property name	Description
ClientLibrary	The connection library type. The value is jConnect for jConnect connections; CT_Library for SAP Open Client connections; None for HTTP connections, and CmdSeq for ODBC, Embedded SQL, OLE DB, ADO.NET, and SQL Anywhere JDBC driver connections.
ClientNodeAddress	The node for the client in a client/server connection. When the client and server are both on the same computer, an empty string is returned. This property is a synonym for the NodeAddress property. The value is *NA* if the request that is currently executing is part of an event handler.
ClientPort	The client's TCP/IP port number or 0 if the connection isn't a TCP/IP connection.
ClientStmtCacheHits	The number of prepares that were not required for this connection because of the client statement cache. This value is the number of additional prepares that would be required if client statement caching was disabled.
ClientStmtCacheMisses	The number of statements in the client statement cache for this connection that were prepared again. This value is the number of times a cached statement was considered for reuse, but could not be reused because of a schema change, a database option setting, or a DROP VARIABLE statement.
close_on_endtrans	Whether cursors are closed at the end of a transaction. This property corresponds to the close_on_endtrans option.
collect_statistics_on_dml_updates	Whether statistics are gathered during the execution of data-altering DML statements such as INSERT, DELETE, and UPDATE. This property corresponds to the collect_statistics_on_dml_updates option.
Commit	The number of Commit requests that have been handled.
CommLink	The communication link for the connection. The value is one of the supported network protocols supported, or *local* for a same-computer connection. The value is *NA* if the request that is currently executing is part of an event handler.
CommNetworkLink	The communication link for the connection. This value returned is one of the supported network protocols. Values include SharedMemory and TCPIP. The value always includes the name of the link, regardless of whether it is same-computer or not. The value is *NA* if the request that is currently executing is part of an event handler.
CommProtocol	The communication protocol. The value is TDS for SAP Open Client and jConnect connections, HTTP for HTTP connections, HTTPS for HTTPS connections. The value is CmdSeq for ODBC, Embedded SQL, OLE DB, ADO.NET, and SQL Anywhere JDBC driver connections.
Compression	Whether communication compression is enabled on the connection. The value is *NA* if the request that is currently executing is part of an event handler.
conn_auditing	Whether auditing is enabled or disabled for the connection when the auditing option is also set to *On*. This property corresponds to the conn_auditing option.

Property name	Description
ConnectedTime	The total length of time, in seconds, that a connection has been connected.
connection_authentication	The string used to authenticate the client. Authentication is required before the database can be modified. This property corresponds to the connection_authentication option.
connection_type	The value of the connection_type database option: one of: Event, Internal, Standard, or Monitor
continue_after_raiserror	Whether execution of a procedure or trigger is stopped whenever the RAISERROR statement is encountered. This property corresponds to the continue_after_raiserror option
conversion_error	Whether data type conversion failures are reported when fetching information from the database. This property corresponds to the value of the conversion_error option.
cooperative_commit_timeout	This property is deprecated. The value of the cooperative_commit_timeout option, which is the time, in milliseconds, that the database server waits for other connections to fill a page of the log before writing to disk.
cooperative_commits	This property is deprecated. The value of the cooperative_commits option, which is On or Off to indicate when commits are written to disk.
CurrentLineNumber	The current line number of the procedure or compound statement a connection is executing. The procedure can be identified using the CurrentProcedure property. If the line is part of a compound statement from the client, an empty string is returned.
CurrentProcedure	The name of the procedure that a connection is currently executing. If the connection is executing nested procedure calls, the name is the name of the current procedure. If there is no procedure executing, an empty string is returned.
Cursor	The number of declared cursors that are currently being maintained by the database server.
CursorOpen	The number of open cursors that are currently being maintained by the database server.
database_authentication	Indicates the string used to authenticate the database. Authentication is required for authenticated database servers before the database can be modified. This property corresponds to the database_authentication option
date_format	The value of the date_format option, which is a string indicating the format for dates retrieved from the database.
date_order	The value of the date_order option, which is a string indicating how dates are formatted.
DBNumber	The ID number of the database.
db_publisher	The user ID of the database publisher. This property corresponds to the db_publisher option.
debug_messages	Whether MESSAGE statements that include a DEBUG ONLY clause are executed. This property corresponds to the debug_messages option
dedicated_task	Whether a request handling task is dedicated exclusively to handling requests for the connection. This property corresponds to the dedicated_task option

Property name	Description
default_dbspace	The name of the default dbspace, or an empty string if the default dbspace has not been specified. This property corresponds to the default_dbspace option,
default_timestamp_increment	The number of microseconds that is added to a column of type TIMESTAMP to keep values in the column unique. This property corresponds to the default_timestamp_increment.
delayed_commit_timeout	The time, in milliseconds, that the database server waits to return control to an application following a COMMIT. This property corresponds to the delayed_commit_timeout option.
delayed_commits	Whether the database server returns control to an application following a COMMIT or not. This property corresponds to the delayed_commits option.
DiskRead	The number of pages that have been read from disk.
DiskReadHint	The number of disk read hints.
DiskReadHintPages	The number of disk read hint pages.
DiskReadIndInt	The number of index internal-node pages that have been read from disk.
DiskReadIndLeaf	The number of index leaf pages that have been read from disk.
DiskReadTable	The number of table pages that have been read from disk.
DiskReadWorkTable	The number of disk work table reads.
disk_sandbox	Whether the read-write file operations of the database are restricted to the directory where the main database file is located. This property corresponds to the disk_sandbox option.
DiskSyncRead	The number of disk reads issued synchronously.
DiskSyncWrite	The number of writes issued synchronously.
DiskWaitRead	The number of times the database server waited for an asynchronous read.
DiskWaitWrite	The number of times the database server waited for an asynchronous write.
DiskWrite	The number of modified pages that have been written to disk.
DiskWriteHint	The number of disk write hints.
DiskWriteHintPages	The number of disk write hint pages.
divide_by_zero_error	Whether if division by zero results in an error (On) or not (Off). This property corresponds to the divide_by_zero_error option.
Encryption	Whether the connection is encrypted or not. The values are None, Simple, rsa_tls, or rsa_tls_fips.
escape_character	This property is reserved for system use. Do not change the setting of this option.

Property name	Description
EventName	The name of the associated event if the connection is running an event handler. Otherwise, an empty string is returned.
exclude_operators	This property is reserved for system use. Do not change the setting of this option.
ExprCacheAbandons	The number of times that the expression cache was abandoned because the hit rate was too low.
ExprCacheDropsToReadOnly	The number of times that the expression cache dropped to read-only status because the hit rate was low.
ExprCacheEvicts	The number of evictions from the expression cache.
ExprCacheHits	The number of hits in the expression cache.
ExprCacheInserts	The number of values inserted into the expression cache.
ExprCacheLookups	The number of lookups done in the expression cache.
ExprCacheResumesOfReadWrite	The number of times that the expression cache resumed read-write status because the hit rate increased.
ExprCacheStarts	The number of times that the expression cache was started.
extern_login_credentials	Whether remote connections are attempted using the logged in user's external login credentials or the effective user's external login credentials. This property corresponds to the extern_login_credentials option.
extended_join_syntax	Whether queries with duplicate correlation name syntax for multi-table joins are allowed, or whether they are reported as errors. This property corresponds to the extended_join_syntax option.
fire_triggers	Whether triggers are fired in the database. This property corresponds to the fire_triggers option.
first_day_of_week	The number that is used for the first day of the week, where 7=Sunday and 1=Monday. This property corresponds to the first_day_of_week option.
for_xml_null_treatment	The value is Omit if elements and attributes that contain NULL values are omitted from the result. The value is Empty if empty elements or attributes are generated for NULL values when the FOR XML clause is used in a query. This property corresponds to the for_xml_null_treatment option.
force_view_creation	This property is reserved for system use. Do not change the setting of this option.
FullCompare	The number of comparisons that have been performed beyond the hash value in an index.
GetData	The number of GETDATA requests.
global_database_id	The starting value used for columns created with DEFAULT GLOBAL AUTOINCREMENT. This property corresponds to the global_database_id option.

Property name	Description
HashForcedPartitions	The number of times that a hash operator was forced to partition because of competition for memory.
HashRowsFiltered	The number of probe rows rejected by bit-vector filters.
HashRowsPartitioned	The number of rows written to hash work tables.
HasSecuredFeature	Whether at least one feature of the feature set is secured (Yes) or not (No). This property has extensions that you can specify when querying the property value.
HashWorkTables	The number of work tables created for hash-based operations.
HeapsCarver	The number of heaps used for short-term purposes such as query optimization.
HeapsLocked	The number of relocatable heaps currently locked in the cache.
HeapsQuery	The number of heaps used for query processing (hash and sort operations).
HeapsRelocatable	The number of relocatable heaps.
http_connection_pool_basesize	The nominal threshold size of database connections. This property corresponds to the http_connection_pool_basesize option.
http_connection_pool_timeout	The maximum length of time that unused connections are stored in the connection pool. This property corresponds to the http_connection_pool_timeout option.
http_session_timeout	The current HTTP session timeout, in minutes. This property corresponds to http_session_timeout option.
HttpServiceName	The service name entry point for the current HTTP request. This property is useful for error reporting and flow control. An empty string is returned when this property is selected from a stored procedure that did not originate from an HTTP request or if the connection is currently inactive or waiting to continue an HTTP session.
IdleTimeout	The idle timeout value of the connection. The value is *NA* if the request that is currently executing is part of an event handler.
IndAdd	The number of entries that have been added to indexes.
IndLookup	The number of entries that have been looked up in indexes.
integrated_server_name	The name of the Domain Controller server used for looking up Microsoft Windows user group membership for integrated logins. This property corresponds to the integrated_server_name option.
IsDebugger	Whether the connection is being used to run the debugger (Yes) or not (No). The value is Yes if the current connection number corresponds to the connection number of a debugger connection, and No otherwise.
isolation_level	The isolation level of the connection. This property corresponds to the isolation_level option.

Property name	Description
java_class_path	The list of additional directories or JAR files that are searched for classes. This property corresponds to the java_class_path option.
java_location	The path of the Java VM for the database if one has been specified. This property corresponds to the java_location option.
java_vm_options	The command line options that the database server uses when it launches the Java VM. This property corresponds to the java_vm_options option.
java_main_userid	This property is deprecated.
Language	The locale language.
LastCommitRedoPos	The redo log position after the last COMMIT operation was written to the transaction log by the connection.
LastIdle	The number of ticks between requests.
LastPlanText	The long text plan of the last query executed on the connection. You control the remembering of the last plan by setting the RememberLastPlan option of the sa_server_option system procedure, or using the -zp server option.
LastReqTime	The time at which the last request for the specified connection started, in the timezone of the database. This property can return an empty string for internal connections, such as events. If the database has the time_zone option set, then the value is returned using the database's time zone.
LastStatement	The most recently prepared SQL statement for the current connection. The LastStatement value is set when a statement is prepared, and is cleared when a statement is dropped. Only one statement string is remembered for each connection. When client statement caching is enabled and a cached statement is reused, the value is an empty string. If sa_conn_activity reports a non-empty value for a connection, it is most likely the statement that the connection is currently executing. If the statement had completed, it would likely have been dropped and the property value would have been cleared. If an application prepares multiple statements and retains their statement handles, then the LastStatement value does not reflect what a connection is currently doing.
LivenessTimeout	The liveness timeout period for the current connection. The value is *NA* if the request that is currently executing is part of an event handler.
lock_rejected_rows	This property is reserved for system use. Do not change the setting of this option.
LockCount	The number of locks held by the connection.

Property name	Description
LockObjectOID	The value is zero if the connection isn't blocked on a table, mutex, or a semaphore, or if the connection is on a different database than the connection calling CONNECTION_PROPERTY. Otherwise, LockObjectOID is the object ID of the table, permanent mutex, or permanent semaphore that the connection is blocked on. A negative value indicates the ID of a temporary mutex or semaphore. LockObjectOID can be used to look up information about temporary mutexes and semaphores using the sp_list_mutexes_semaphores system procedure. If the object is a table, LockObjectOID can be used to look up table information using the SYSTAB system view.
LockObjectType	The ID for the type of object the connection is blocked on. Use the ID to look up the object type in the SYSOBJECT view. Can be one of 'TABLE' or 'MUTEX SEMAPHORE'.
LockIndexID	The identifier of the locked index.
LockName	The 64-bit unsigned integer value representing the lock for which a connection is waiting.
LockRowID	The identifier of the locked row.
LockTableOID	Zero if the connection isn't blocked, or isn't blocked on a table, or if the connection is on a different database than the connection calling CONNECTION_PROPERTY. Otherwise, this is the object ID of the table for the lock on which this connection is waiting. The object ID can be used to look up table information using the SYSTAB system view.
log_deadlocks	Whether deadlock information is recorded (On) or not (Off). This property corresponds to the log_deadlocks option.
LogFreeCommit	The number of redo free commits. A redo free commit occurs when a commit of the transaction log is requested but the log has already been written (so the commit was done for free.)
login_mode	The type of login that is supported. This property corresponds to the login_mode option.
login_procedure	The name of the stored procedure used to set compatibility options at startup. This property corresponds to the login_procedure option.
LoginTime	The date and time the connection was established. If the database has the time_zone option set, then the value is returned using the database's time zone.
LogWrite	The number of pages that have been written to the transaction log.
materialized_view_optimization	The value of the materialized_view_optimization option for the connection, which indicates whether materialized views are used during query optimization.
max_connections	The value of the max_connections option, which indicates the number of concurrent connections allowed to the database.
max_client_statements_cached	The value of the max_client_statements_cached option, which indicates the number of statements cached by the client.
max_cursor_count	The maximum number of cursors that a connection can use at once. This property corresponds to the max_cursor_count option.
max_hash_size	This property is deprecated.

Property name	Description
max_plans_cached	The maximum number of execution plans to be stored in a cache. This property corresponds to the max_plans_cached option.
max_priority	The maximum priority level a connection can have. This property corresponds to the max_priority option for the connection.
max_query_tasks	The maximum number of requests that the database server can use to process a query. This property corresponds to the max_query_tasks option.
max_recursive_iterations	The maximum number of iterations a recursive common table expression can make. This property corresponds to the max_recursive_iterations option.
max_statement_count	The maximum number of prepared statements that a connection can use simultaneously. This property corresponds to max_statement_count option.
max_temp_space	The maximum amount of temporary file space available for a connection. This property corresponds to the max_temp_space option for the connection.
MessageReceived	The string that was generated by the MESSAGE statement that caused the WAITFOR statement to be interrupted. Otherwise, an empty string is returned.
min_password_length	The minimum length for new passwords in the database. This property corresponds to min_password_length option.
min_role_admins	The minimum number of administrators required for a role. This property corresponds to the min_role_admins option.
Name	The name of the current connection. You can specify a connection name using the ConnectionName (CON) connection parameter.
NcharCharSet	The NCHAR character set used by the connection. This property has extensions that you can specify when querying the property value.
nearest_century	The value of the nearest_century option, which indicates how two-digit years are interpreted in string-to-date conversions.
NodeAddress	The node for the client in a client/server connection. When the client and server are both on the same computer, an empty string is returned.
non_keywords	The value of the non_keywords option, which is a list of keywords, if any, that are turned off so they can be used as identifiers.
NumLocalTempTables	The number of local temporary tables in use by the connection.
Number	The connection ID (a number) for the current connection.

Property name	Description
odbc_describe_binary_as_varbinary	The value is Off if the SQL Anywhere ODBC driver describes both BINARY and VARBINARY columns as SQL_BINARY. The value is On if the ODBC driver describes BINARY and VARBINARY columns as SQL_VARBINARY. This property corresponds to the odbc_describe_binary_as_varbinary option.
odbc_distinguish_char_and_varchar	Whether CHAR columns are described as SQL_CHAR (On) or they are described as SQL_VARCHAR (OFF). This property corresponds to the odbc_distinguish_char_and_varchar option.
oem_string	The string stored in the header page of the database file. This property corresponds to the oem_string option.
on_charset_conversion_failure	The behavior when an error is encountered during character set conversion. This property corresponds to the on_charset_conversion_failure option.
on_tsql_error	The behavior when an error is encountered while executing a stored procedure or T-SQL batch. This property corresponds to the on_tsql_error option.
optimization_goal	How query processing is optimized. This property corresponds to the optimization_goal option.
optimization_level	The value of the optimization_level option, which is a value between 0 and 15. This number is used to control the level of effort made by the SQL Anywhere query optimizer to find an access plan for a SQL statement.
optimization_workload	The level of effort made by the SQL Anywhere query optimizer to find an access plan for a SQL statement. This property corresponds to the optimization_workload option for the connection.t
OSUser	The operating system user name associated with the client process. If the client process is impersonating another user (or the set ID bit is set on UNIX/Linux), the impersonated user name is returned. An empty string is returned for version 10.0.1 and earlier clients, and for HTTP and TDS clients.
PacketSize	The packet size used by the connection, in bytes. The value is *NA* if the request that is currently executing is part of an event handler. This property corresponds to the CommBufferSize (CBSIZE) connection parameter.
PacketsReceived	The number of client/server communication packets received. This value is not updated for HTTP or HTTPS connections.
PacketsReceivedUncomp	The number of packets that would have been received during client/server communications if compression was disabled. (This value is the same as the value for PacketsReceived if compression is disabled.)
PacketsSent	The number of client/server communication packets sent. This value is not updated for HTTP or HTTPS connections.
PacketsSentUncomp	The number of packets that would have been sent during client/server communications if compression was disabled. (This value is the same as the value for PacketsSent if compression is disabled.)

Property name	Description
parameterization_level	The value of the parameterization_level option for the connection, which indicates the statement parameterization behavior.
ParameterizationPrepar-eCount	The number of prepares for statements that have been automatically parameterized.
ParentConnection	The connection ID of the connection that created a temporary connection to perform a database operation (such as performing a backup or creating a database). For other types of connections, the value is NULL.
pinned_cursor_per-cent_of_cache	The value of the pinned_cursor_percent_of_cache option, which indicates the percentage of the cache that can be used for pinning cursors.
post_login_procedure	The name of the procedure whose result set contains messages that should be displayed by applications when a user connects. This property corresponds to the post_login_procedure option.
precision	The value of the precision option, which indicates the decimal and numeric precision setting.
prefetch	The value of the prefetch option. The value is Off if no prefetching is done. The value is Conditional if prefetching occurs unless the cursor type is SENSITIVE or the query includes a proxy table. The value is Always if prefetching is done even for SENSITIVE cursor types and cursors that involve a proxy table.
Prepares	The number of statement preparations performed for the connection.
PrepStmt	The number of prepared statements currently being maintained by the database server for this connection.
preserve_source_format	Whether the original source definition of procedures, triggers, views, and event handlers is saved in system tables (On) or not (Off). This property corresponds to the preserve_source_format option.
prevent_arti-cle_pkey_update	Whether updates are not allowed to the primary key columns of tables involved in publications (On) or not (Off). This property corresponds to the prevent_article_pkey_update option.
priority	The value of the priority option for the connection, which indicates the priority level of a connection.
Progress	Information about how long a query has been running. This property has extensions that you can specify when querying the property value.
progress_messages	The value of the progress_messages option.
query_mem_timeout	The value of the query_mem_timeout option.
QueryBypassed	The number of requests optimized by the optimizer bypass.
QueryBypassedCosted	The number of requests processed by the optimizer bypass using costing.

Property name	Description
QueryBypassedHeuristic	The number of requests processed by the optimizer bypass using heuristics.
QueryBypassedOptimized	The number of requests initially processed by the optimizer bypass and subsequently fully optimized by the optimizer.
QueryCachedPlans	The number of query execution plans currently cached for the connection.
QueryCachePages	The number of cache pages used to cache execution plans.
QueryDescribedBypass	The number of describe requests processed by the optimizer bypass.
QueryDescribedOptimizer	The number of describe requests processed by the optimizer.
QueryHeapPages	The number of cache pages used for query processing (hash and sort operations).
QueryJHToJNLOptUsed	The number of times a hash join was converted to a nested loop join.
QueryLowMemoryStrategy	The number of times the server changed its execution plan during execution as a result of low memory conditions. The strategy can change because less memory is currently available than the optimizer estimated, or because the execution plan required more memory than the optimizer estimated.
QueryMemActiveCurr	The number of requests actively using query memory.
QueryMemGrantFailed	The total number of times a request waited for query memory, but failed to get it.
QueryMemGrantGranted	The number of pages currently granted to requests.
QueryMemGrantRequested	The total number of times any request attempted to acquire query memory.
QueryMemGrantWaited	The total number of times any request waited for query memory.
QueryMemGrantWaiting	The current number of requests waiting for query memory.
QueryOpened	The number of OPEN requests for execution.
QueryOptimized	The number of requests fully optimized.
QueryReused	The number of requests that have been reused from the plan cache.
QueryRowsFetched	The number of rows that have been read from base tables, either by a sequential scan or an index scan, for this connection.
QueryRowsMaterialized	The number of rows that are written to work tables during query processing.

Property name	Description
quoted_identifier	Whether strings enclosed in double quotes are interpreted as identifiers (On), or if they are interpreted as literal strings (Off). This property corresponds to the quoted_identifier option.
read_past_deleted	Whether sequential scans at isolation levels 1 and 2 skip uncommitted deleted rows (On), or sequential scans block on uncommitted deleted rows at isolation levels 1 and 2 (Off). This property corresponds to the read_past_deleted option.
recovery_time	The maximum length of time, in minutes, that the database server will take to recover from system failure. This property corresponds to the recovery_time option.
RecursiveIterations	The number of iterations for recursive unions.
RecursiveIterationsHash	The number of times recursive hash join used a hash strategy.
RecursiveIterations-Nested	The number of times recursive hash join used a nested loop strategy.
RecursiveJNLMisses	The number of index probe cache misses for recursive hash join.
RecursiveJNLProbes	The number of times recursive hash join attempted an index probe.
remote_idle_timeout	The time, in seconds, of inactivity that web service client procedures and functions will tolerate. This property corresponds to the remote_idle_timeout option.
replicate_all	For internal use only.
ReqCountActive	The number of requests processed, or NULL if the RequestTiming server property is set to Off.
ReqCountBlockContention	The number of times the connection waited for atomic access, or NULL if the -zt option was not specified.
ReqCountBlockIO	The number of times the connection waited for I/O to complete, or NULL if the -zt option was not specified.
ReqCountBlockLock	The number of times the connection waited for a lock, or NULL if the -zt option was not specified.
ReqCountUnscheduled	The number of times the connection waited for scheduling, or NULL if the -zt option was not specified.
ReqStatus	The status of the request. The value is Idle when the connection is not currently processing a request. The value is Unscheduled* when the connection has work to do and is waiting for an available database server worker. The value is BlockedIO* when the connection is blocked waiting for an I/O. The value is BlockedContention* when the connection is blocked waiting for access to shared database server data structures. The value is BlockedLock when the connection is blocked waiting for a locked object. The value is Executing when the connection is executing a request. The values marked with an asterisk (*) are only returned when logging of request timing information has been turned on for the database server using the -zt server option. If request timing information is not being logged (the default), the values are reported as Executing.

Property name	Description
ReqTimeActive	The amount of time in seconds spent processing requests, or NULL if the -zt option was not specified.
ReqTimeBlockContention	The amount of time in seconds spent waiting for atomic access, or NULL if the RequestTiming server property is set to Off.
ReqTimeBlockIO	The amount of time in seconds spent waiting for I/O to complete, or NULL if the -zt option was not specified.
ReqTimeBlockLock	The amount of time in seconds spent waiting for a lock, or NULL if the -zt option was not specified.
ReqTimeUnscheduled	The amount of unscheduled time in seconds, or NULL if the -zt option was not specified.
ReqType	The type of the last request. If a connection has been cached by connection pooling, its ReqType value is CONNECT_POOL_CACHE.
request_timeout	The value of the request_timeout option, which indicates the maximum time a single request can run.
RequestsReceived	The number of client/server communication requests or round trips. It is different from PacketsReceived in that multi-packet requests count as one request, and liveness packets are not included.
reserved_connections	The number of connections that are reserved for standard connections. This property corresponds to the reserved_connections option.
reserved_keywords	The value of the reserved_keywords option, which specifies a list of non-default reserved keywords that are enabled for the database.
return_date_time_as_string	Whether DATE, TIME, and TIMESTAMP values are returned to applications as a string (On), or they are returned as a DATE, TIME, or TIMESTAMP data type (Off). This property corresponds to the return_date_time_as_string option.
Rlbk	The number of rollback requests that have been handled.
rollback_on_deadlock	Whether transaction are automatically rolled back if it encounters a deadlock (On) or not (Off). This property corresponds to the rollback_on_deadlock option.
RollbackLogPages	The number of pages in the rollback log.
row_counts	Whether the row count is always accurate (On), or the row count is usually an estimate (Off). This property corresponds to the row_counts option.
scale	The decimal and numeric scale for the connection. This property corresponds to the scale option.

Property name	Description
secure_feature_key	This property is deprecated. The value of the secure_feature_key option, which stores the key that is used to enable and disable features for a database server. Selecting the value of this property always returns an empty string.
ServerNodeAddress	The node for the server in a client/server connection. When the client and server are both on the same computer, an empty string is returned. The value is *NA* if the request that is currently executing is part of an event handler.
ServerPort	The database server's TCP/IP port number or 0.
SessionCreateTime	The time the HTTP session was created. If the database has the time_zone option set, then the value is returned using the database's time zone.
SessionID	The session ID for the connection if it exists, otherwise, an empty string.
SessionLastTime	The time of the last request for the HTTP session. If the database has the time_zone option set, then the value is returned using the database's time zone.
SessionTimeout	The time, in minutes, the HTTP session persists during inactivity.
SnapshotCount	The number of snapshots associated with the connection.
sort_collation	The value of the sort_collation option. The value is Internal if the ORDER BY clause remains unchanged; otherwise, the value is the collation name or the collation ID.
SortMergePasses	The number of merge passes used during sorting.
SortRowsMaterialized	The number of rows written to sort work tables.
SortRunsWritten	The number of sorted runs written during sorting.
SortSortedRuns	The number of sorted runs created during run formation.
SortWorkTables	The number of work tables created for sorting.
sql_flagger_error_level	The value of the sql_flagger_error_level option, which controls the response to any SQL that is not part of the specified standard. This property corresponds to the sql_flagger_error_level option.
sql_flagger_warning_level	The value of the sql_flagger_warning_level. This property corresponds to the sql_flagger_warning_level option.
st_geometry_asbinary_format	How spatial values are converted from a geometry to a binary format. This property corresponds to the st_geometry_asbinary_format option.
st_geometry_astext_format	How spatial values are converted from a geometry to text. This property corresponds to the st_geometry_astext_format option.
st_geometry_asxml_format	How spatial values are converted from a geometry to XML. This property corresponds to the st_geometry_asxml_format option.

Property name	Description
st_geometry_describe_type	How spatial values are described. This property corresponds to the st_geometry_describe_type option.
st_geometry_interpolation	The interpolation setting for ST_CircularString geometries. This property corresponds to st_geometry_interpolation option.
st_geometry_on_invalid	The behavior when a geometry fails surface validation. This property corresponds to the st_geometry_on_invalid option.
StatementDescribes	The total number of statements processed by DESCRIBE requests.
StatementPostAnnotates	The number of statements processed by the semantic query transformation phase.
StatementPostAnnotatesSimple	The number of statements processed by the semantic query transformation phase, but that skipped some of the semantic transformations.
StatementPostAnnotatesSkipped	The number of statements that have completely skipped the semantic query transformation phase.
string_rtruncation	Whether an error is raised when a string is truncated (On), or no error is not raised (Off) This property corresponds to the string_rtruncation option.
subsume_row_locks	Whether the database server acquires individual row locks for a table (On), or not (Off). This property corresponds to the subsume_row_locks option.
suppress_tds_debugging	Whether TDS debugging information appears in the database server messages window (Off), or the debugging information does not appear in the database server messages window (On). This property corresponds to the suppress_tds_debugging option.
synchronize_mirror_on_commit	Whether the database mirror server is synchronized on commit (On) or not (Off). This property corresponds to the synchronize_mirror_on_commit option.
tds_empty_string_is_null	Whether empty strings are returned as NULL for TDS connections (On), or if a string containing one blank character is returned for TDS connections (Off). This property corresponds to the tds_empty_string_is_null option.
temp_space_limit_check	Whether the database server checks the amount of temporary space available for a connection (On), or the database server does not check the amount of space available for a connection (Off). This property corresponds to the temp_space_limit_check option.
TempFilePages	The number of temporary file pages used by the connection.
TempTablePages	The number of pages in the temporary file used for temporary tables.
time_format	The string format used for times retrieved from the database. This property corresponds to the time_format option.

Property name	Description
time_zone	The time zone that the database uses for time zone calculations. This property corresponds to the time_zone option.
time_zone_adjustment	The number of minutes that must be added to the Coordinated Universal Time (UTC) to display time local to the connection. This property corresponds to the time_zone_adjustment option.
timestamp_format	The format for timestamps that are retrieved from the database. This property corresponds to the timestamp_format option.
time-stamp_with_time_zone_ format	The format for TIMESTAMP WITH TIME ZONE values retrieved from the database. This property corresponds to the timestamp_with_time_zone_format option.
TimeZoneAdjustment	The number of minutes that must be added to the Coordinated Universal Time (UTC) to display time local to the connection.
TransactionStartTime	The value is a string containing the time the database was first modified after a COMMIT or ROLLBACK, or an empty string if no modifications have been made to the database since the last COMMIT or ROLLBACK. If the database has the time_zone option set, then the value is returned using the database's time zone.
truncate_time-stamp_values	Whether the number of decimal places used in the TIMESTAMP values is limited (On) or not (Off). This property corresponds to the truncate_timestamp_values option.
trusted_certificates_file	The file that contains the list of trusted Certificate Authority certificates when the database server acts as a client to an LDAP server. This property corresponds to the trusted_certificates_file option.
tsql_outer_joins	Whether Transact-SQL outer joins can be used in DML statement (On) or not (Off). This property corresponds to the value of the tsql_outer_joins option.
tsql_variables	Whether you can use the @ sign instead of the colon as a prefix for host variable names in Embedded SQL (On) or not (Off). This property corresponds to the value of the tsql_variables option.
UncommitOp	The number of uncommitted operations.
updatable_state-ment_isolation	The isolation level (0, 1, 2, or 3) used by updatable statements when the isolation_level option is set to Readonly-statement-snapshot. This property corresponds to the updatable_statement_isolation option.
update_statistics	Whether the connection can send query feedback to the statistics governor (On) or the statistics governor does not receive query feedback from the current connection (Off). This property corresponds to the update_statistics option.
upgrade_database_capability	This property is reserved for system use. Do not change the setting of this option.

Property name	Description
user_estimates	The value that controls whether selectivity estimates in query predicates are respected or ignored by the query optimizer. This property corresponds to the user_estimates option.:
UserAppInfo	The string specified by the AppInfo connection parameter in a connection string.
UserDefinedCounterRate01	The current value of the user-defined performance counter. The semantics of this property are defined by the client application.
UserDefinedCounterRate02	The current value of the user-defined performance counter. The semantics of this property are defined by the client application.
UserDefinedCounterRate03	The current value of the user-defined performance counter. The semantics of this property are defined by the client application.
UserDefinedCounterRate04	The current value of the user-defined performance counter. The semantics of this property are defined by the client application.
UserDefinedCounterRate05	The current value of the user-defined performance counter. The semantics of this property are defined by the client application.
UserDefinedCounterRaw01	The current value of the user-defined performance counter. The semantics of this property are defined by the client application.
UserDefinedCounterRaw02	The current value of the user-defined performance counter. The semantics of this property are defined by the client application.
UserDefinedCounterRaw03	The current value of the user-defined performance counter. The semantics of this property are defined by the client application.
UserDefinedCounterRaw04	The current value of the user-defined performance counter. The semantics of this property are defined by the client application.
UserDefinedCounterRaw05	The current value of the user-defined performance counter. The semantics of this property are defined by the client application.
UserID	The user ID for the connection.
UtilCmdsPermitted	Whether SQL statements such as CREATE DATABASE, DROP DATABASE, and RESTORE DATABASE are permitted for the connection or not. The value is an empty string if the specified connection ID is not for the current connection.
uuid_has_hyphens	The format of unique identifier values when they are converted to strings. When the option is set to On, the resulting strings contain four hyphens. This property corresponds to the uuid_has_hyphens option.
verify_password_function	The name of the function used for password verification if one has been specified. This property corresponds to the verify_password_function.

Property name	Description
wait_for_commit	Whether the database does not check foreign key integrity until the next COMMIT statement (On), or all foreign keys that are not created with the CHECK ON COMMIT clause are checked as they are inserted, updated or deleted (Off). This property corresponds to the wait_for_commit option.
WaitStartTime	The time at which the connection started waiting (or an empty string if the connection is not waiting). If the database has the time_zone option set, then the value is returned using the database's time zone.
WaitType	The reason for the wait, if it is available. The value is lock when the connection is waiting on a lock The value is waitfor when the connection is executing a WAITFOR statement. The value is an empty-string when the connection is not waiting, or when the reason for the wait is not available.
webservice_namespace_host	The hostname to be used as the XML namespace within generated WSDL documents if one has been specified. This property corresponds to the webservice_namespace_host option,
webservice_sessionid_name	The session identifier name that is used by the web server to determine whether session management is being used. This property corresponds to the webservice_sessionid_name option.

Related Information

Alphabetical List of Connection Parameters [page 45]
-ze Database Server Option [page 533]
xp_getenv System Procedure
CONNECTION_PROPERTY Function [System]
sa_conn_properties System Procedure
List of Database Properties [page 917]
List of Database Server Properties [page 900]

1.5.7.2 List of Database Server Properties

Database server properties are available for each connection to a database. Use the PROPERTY system function to retrieve the value for an individual property and use the sa_eng_properties system procedure to retrieve the values of all database server properties.

Example

The following statement returns the number of cache pages used for global server data structures:

```
SELECT PROPERTY ( 'MainHeapPages' );
```

Use the sa_eng_properties system procedure to retrieve the values of all database server properties:

```
CALL sa_eng_properties;
```

Database Server Properties

Property name	Description
ActiveReq	The number of server workers that are currently handling client-side requests.
ApproximateCPUTime	An estimate of the amount of CPU time accumulated by the database server, in seconds. The value may differ from the actual value by as much as 50 percent, although typical variations are in the 5 to 10 percent range. On multi-processor computers, each CPU (or hyperthread or core) accumulates time, so the sum of accumulated times for all connections may be greater than the elapsed time.
AutoMultiProgrammingLevel	Whether the database server is automatically adjusting its multiprogramming level.
AutoMultiProgrammingLevelStatistics	Whether messages about automatic adjustments to the database server's multiprogramming level are displayed in the database server message log.
AvailIO	The current number of available I/O control blocks.
BuildChange	Reserved for system use.
BuildClient	Reserved for system use. Do not change the setting of this property.
BuildProduction	Whether the database server is compiled for production use (Yes) or whether the database server is a debug build (No)..
BuildReproducible	Reserved for system use.
BytesReceived	The number of bytes received during client/server communications. This value is updated for HTTP and HTTPS connections.
BytesReceivedUncomp	The number of bytes that would have been received during client/server communications if compression was disabled. (This value is the same as the value for BytesReceived if compression is disabled.)
BytesSent	The number of bytes sent during client/server communications. This value is updated for HTTP and HTTPS connections.
BytesSentUncomp	The number of bytes that would have been sent during client/server communications if compression was disabled. (This value is the same as the value for BytesSent if compression is disabled.)
CacheAllocated	The number of cache pages that have been allocated for purposes other than file caching.

Property name	Description
CacheFile	The number of cache pages used to hold data from database files.
CacheFileDirty	The number of cache pages that are dirty (needing a write).
CacheFree	The number of cache pages not being used.
CacheHits	The number of database page lookups.
CacheMisses	The number of times a page was not in the cache.
CachePanics	The number of times the cache manager has failed to find a page to allocate.
CachePinned	The number of pinned cache pages.
CacheRead	The number of cache reads.
CacheReplacements	The number of pages in the cache that have been replaced.
CacheScavenges	The number of times the cache manager has scavenged for a page to allocate.
CacheScavengeVisited	The number of pages visited while scavenging for a page to allocate.
CacheSizingStatistics	Whether the database server is displaying cache sizing statistics when the cache is resized.
CarverHeapPages	The number of heap pages used for short-term purposes, such as query optimization.
CharSet	The CHAR character set in use by the database server.
ClientStmtCacheHits	The number of prepares that were not required because of the client statement cache. This is the number of additional prepares that would be required if client statement caching was disabled.
ClientStmtCacheMisses	The number of statements in the client statement cache that were prepared again. This is the number of times a cached statement was considered for reuse, but could not be reused because of a schema change, a database option setting, or a DROP VARIABLE statement.
CockpitDB	The set of options currently being used by the database server, as well as the Temp parameter. When Temp is set to yes, the Cockpit database is a temporary database.
CockpitURL	A semicolon-delimited list of valid URLs for the Cockpit.
CollectStatistics	Whether the database server is collecting performance statistics.
CommandLine	The command line arguments that were used to start the database server. If the encryption key for a database was specified using the -ek option, the key is replaced with a constant string of asterisks in the value for this property. If the utility database user ID and password was specified using the -su option, they are replaced with a constant string of asterisks in the value for this property.
Commit	The number of Commit requests that have been handled.

Property name	Description
CompactPlatformVer	A condensed version of the PlatformVer property.
CompanyName	The name of the company owning this software.
CompletedReq	The number of requests that have been completed.
ConnCount	The number of connections to the database server. This property value does not include connections used for internal operations, but does include connections used for events and external environment support.
ConnectedTime	The total length of time, in seconds, that all connections have been connected to the database server. The value is updated only when a request completes for a connection or when a connection disconnects. As a result, the value can lag behind for connections that are idle or busy executing for a long time in the database server. The value includes time accrued by any connection, including database events and background server connections (such as the database cleaner).
ConnsDisabled	Whether the server option is set to disable new connections.
ConsoleLogFile	The name of the file where database server messages are logged if the -o option was specified. If the -option was not specified, the value is an empty string.
ConsoleLogMaxSize	The maximum size in bytes of the file used to log database server messages.
CurrentCacheSize	The current cache size, in kilobytes.
CurrentMirrorBackgroundWorkers	The number of workers currently being used for database mirroring background tasks. These workers are separate from those controlled by the multiprogramming level.
CurrentMultiProgrammingLevel	The current number of tasks that the database server can process concurrently.
CurrRead	The current number of file reads that were issued by the database server, but that have not completed yet.
CurrWrite	The current number of file writes that were issued by the database server, but that have not completed yet.
Cursor	The number of declared cursors that are currently being maintained by the database server.
CursorOpen	The number of open cursors that are currently being maintained by the database server.
DebuggingInformation	Whether the server is displaying diagnostic messages for troubleshooting.
DefaultCollation	The collation that would be used for new databases if none is explicitly specified.
DefaultNcharCollation	The name of the default NCHAR collation on the server computer (UCA if ICU is installed, and UTF8BIN otherwise).

Property name	Description
DiskRead	The number of disk reads.
DiskReadHintScatterLimit	The imposed limit on the size (in bytes) of a scatter read hint.
DiskRetryRead	The number of disk read retries.
DiskRetryReadScatter	The number of disk read retries for scattered reads.
DiskRetryWrite	The number of disk write retries.
DiskSandbox	Whether the read-write file operations of the database are restricted to the directory where the main database file is located (On) or not (Off).
DiskWrite	The number of modified pages that have been written to disk.
DropBadStatistics	Automatic statistics management to drop statistics that return bad estimates.
DropUserStatistics	Automatic statistics management to drop statistics that have not been used for 90 consecutive days.
EventTypeDesc	The description of the event type associated with a given event type ID.
EventTypeName	The system event type name associated with a given event type ID.
ExchangeTasks	The number of tasks currently being used for parallel execution of queries.
ExchangeTasksCompleted	The total number of internal tasks that have been used for intra-query parallelism since the database server started.
FipsMode	Whether the -fips option was specified when the database server was started.
FirstOption	The number that represents the first connection property that corresponds to a database option.
FreeBuffers	The number of available network buffers.
FunctionMaxParms	The maximum number of parameters that can be specified by a function. The function is identified by the value specified by the `function-number`, which is a positive integer. For example: `SELECT PROPERTY ( 'FunctionMaxParms', function-number );` The `function-number` is subject to change between releases.
FunctionMinParms	The minimum number of parameters that must be specified by a function. The function is identified by the value specified by the `function-number`, which is a positive integer. For example: `SELECT PROPERTY ( 'FunctionMinParms', function-number );` The `function-number` is subject to change between releases.

Property name	Description
FunctionName	The name of the function identified by the value specified by the `function-number` (which is a positive integer): `SELECT PROPERTY ( 'FunctionName', function-number );` The `function-number` is subject to change between releases.
HasSecuredFeature	Whether at least one feature of the all feature set is secured at the global server level. This property has extensions that you can specify when querying the property value.
HasSecureFeatureKey	Whether the database server has at least one secure feature key. This property has extensions that you can specify when querying the property value.
HeapsCarver	The number of heaps used for short-term purposes such as query optimization.
HeapsLocked	The number of relocatable heaps currently locked in the cache.
HeapsQuery	The number of heaps used for query processing (hash and sort operations).
HeapsRelocatable	The number of relocatable heaps.
HttpAddresses	A semicolon-delimited list of the IP addresses that the database server is listening to for HTTP connections from clients. For example: `(::1):80;127.0.0.1:80`
HttpConnectionsQueued	The number of connections that are currently in the queue.
HttpListeners	A semicolon-delimited list of `IP address:port` pairs that the database server is using to listen for HTTP connections.
HttpNumActiveReq	The number of HTTP connections that are actively processing an HTTP request. An HTTP connection that has sent its response is not included.
HttpNumConnections	The number of HTTP connections that are currently open within the database server. They may be actively processing a request or waiting in a queue of long lived (keep-alive) connections.
HttpNumSessions	The number of active and dormant HTTP sessions within the database server.
HttpPorts	The HTTP port numbers for the web server as a comma-delimited list.
HttpQueueCount	The total number of connections that have been queued since the database server started.
HttpQueueMaxCount	The maximum number of connections that have been in the queue at one time.
HttpQueueTimedOut	The total number of connections that have timed out after sitting in the queue.

Property name	Description
HttpsAddresses	A semicolon-delimited list of the IP addresses that the server is listening to for HTTPS connections from clients. For example: `(::1):443;127.0.0.1:443`
HttpsListeners	A semicolon-delimited list of `IP address:port` pairs that the database server is using to listen for HTTPS connections.
HttpsNumActiveReq	The number of secure HTTPS connections that are actively processing an HTTPS request. An HTTPS connection that has sent its response is not included.
HttpsNumConnections	The number of HTTPS connections that are currently open within the database server. They may be actively processing a request or waiting in a queue of long lived (keep-alive) connections.
HttpsPorts	The HTTPS port numbers for the web server as a comma-delimited list.
IdleTimeout	The default idle timeout.
IPAddressMonitorPeriod	The time in seconds that a database server checks for new IP addresses.
IsAesniAvailable	Whether the database server computer's CPU supports the Intel AES-NI instruction set and the computer is running a supported operating system.
IsFipsAvailable	Whether the FIPS-certified DLL is installed.
IsIQ	Reserved for system use.
IsNetworkServer	Whether the connection is to a network database server (Yes), or to a personal database server (No).
IsPortableDevice	Whether the database server is running on a laptop, notebook, or other portable device. VMware is not taken into account, so the value is No for a database server running on a VM that is running on a laptop. On Windows, if it is not possible to determine whether the device is portable, the value is NULL. This property is always NULL on UNIX and Linux systems.
IsRsaAvailable	Whether the RSA DLL is installed.
IsRuntimeServer	This property is No for all versions of the database server.
IsService	Whether the database server is running as a service.
JavaVM	An empty string if the database server uses one Java VM per database. If the database server uses one Java VM for all databases, this property indicates the path to the JAVA executable.
Language	The locale language for the server.

Property name	Description
LastConnectionProperty	The number that represents the last connection property.
LastDatabaseProperty	The number that represents the last database property.
LastOption	The number that represents the last connection property that corresponds to a database option.
LastServerProperty	The number that represents the last server property.
LegalCopyright	The copyright string for the software.
LegalTrademarks	The trademark information for the software.
LicenseCount	The number of licensed seats or processors.
LicensedCompany	The name of the licensed company.
LicensedUser	The name of the licensed user.
LicenseKey	Reserved for system use.
LicenseType	The license type. Can be networked seat (per-seat) or CPU-based.
LivenessTimeout	The client liveness timeout default.
LockedCursorPages	The number of pages used to keep cursor heaps pinned in memory.
LockedHeapPages	The number of heap pages locked in the cache.
LockTableContention-Count	The number of times that significant contention has occurred on any of the lock tables. A value larger than the number of hours the database has been running indicates a need to increase the number of lock tables using the -lt switch.
MachineName	The name of the computer running a database server. Typically, this is the computer's host name.
MainHeapBytes	The amount of memory allocated for server data structures, in bytes.
MainHeapPages	The amount of memory allocated for server data structures, in database pages.
MapPhysicalMemoryEng	The number of database page address space windows mapped to physical memory in the cache using Address Windowing Extensions.
MaxCacheSize	The maximum cache size allowed, in kilobytes.
MaxConnections	The maximum number of concurrent connections that the database server allows. For the network database servers the default depends upon your license. The default value can be lowered using the -gm server option. For the personal database server, the default value is 10.
MaxEventType	The maximum valid event type ID.

Property name	Description
MaxMessage	This property is deprecated. The current maximum line number that can be retrieved from the database server messages window. This represents the most recent message displayed in the database server messages window.
MaxMirrorBackground-Workers	The highest number of workers used for database mirroring background tasks since the server started. These workers are separate from those controlled by the multiprogramming level.
MaxMultiProgrammingLe-vel	The maximum number of tasks that the database server can process concurrently. When AutoMultiProgrammingLevel is enabled, the server may increase the multiprogramming level up to this value.
MaxRemoteCapability	The maximum valid capability ID.
Message,**linenumber**	A line from the database server messages window, prefixed by the date and time the message appeared. The second parameter specifies the line number. This property is deprecated. The value for `PROPERTY( "message" )` is the first line of output that was written to the database server messages window. Calling `PROPERTY( "message", n )` The *nth* line of server output (with zero being the first line). The buffer is finite, so as messages are generated, the first lines are dropped and may no longer be available in memory. In this case, the value is NULL.
MessageCategoryLimit	The minimum number of messages of each severity and category that can be retrieved using the sa_server_messages system procedure. The default value is 400.
MessageText, **linenumber**	This property is deprecated. The text associated with the specified line number in the database server messages window, without a date and time prefix. The second parameter specifies the line number.
MessageTime, **linenumber**	This property is deprecated. The date and time associated with the specified line number in the database server messages window. The second parameter specifies the line number.
MessageWindowSize	This property is deprecated. The maximum number of lines that can be retrieved from the database server messages window.
MinCacheSize	The minimum cache size allowed, in kilobytes.
MiniDumpType	The current setting for the crash dump type.
MinMultiProgrammingLe-vel	The minimum number of tasks that the server can process concurrently. When AutoMultiProgrammingLevel is enabled, the server may decrease the multiprogramming level down to this value.
MultiPacketsReceived	The number of multi-packet requests received during client/server communications.
MultiPacketsSent	The number of multi-packet responses sent during client/server communications.
MultiPageAllocs	The number of multi-page cache allocations.

Property name	Description
MultiProgrammingLevel	The current maximum number of concurrent tasks the server will process simultaneously. Requests are queued if there are more concurrent tasks than this value. This can be changed with the -gn server option.
Name	The alternate name of the server used to connect to the database if one was specified, otherwise, The real server name. If the client is connected to a copy node and specified NodeType=COPY in the connection string, then the value of this property may be different from the database server name specified in the client connection string by the ServerName connection parameter.
NativeProcessorArchitecture	A string that identifies the native processor type on which the software is running. For platforms where a processor can be emulated (such as x86 on x64), the actual processor type - not the OS architecture type - is returned. The value does not indicate whether the operating system is 32-bit or 64-bit. The following are possible values for the indicated operating system. **Windows:** X86, X86_64 **Linux** X86, X86_64, ARM, ARM_64 **macOS:** X86_64 **Solaris:** SPARC, X86_64 **AIX:** PPC **HP-UX:** IA64 X86 represents a 32-bit hardware architecture. X86_64 represents a 64-bit hardware architecture.
NumLockTables	The number of internal lock tables used by the database. These are internal structures that are not visible to users. The server normally selects an appropriate number of lock tables.
NumLogicalProcessors	The number of logical processors (including cores and hyperthreads) enabled on the server computer.
NumLogicalProcessorsUsed	The number of logical processors the database server will use. On Windows, use the -gtc option to change the number of logical processors used.
NumPhysicalProcessors	The number of physical processors enabled on the server computer. This value is NumLogicalProcessors divided by the number of cores or hyperthreads per physical processor. On some non-Windows platforms, cores or hyperthreads may be counted as physical processors.
NumPhysicalProcessorsUsed	The number of physical processors the database server will use. On Windows, you can use the -gt option to change the number of physical processors used by the network database server. The personal server is limited to four cores on one processor on some platforms.
ObjectType	The type of database object. This value is used by the SYSOBJECT system view.
ODataAddresses	A semicolon-delimited list of the TCP/IP addresses and ports that the OData server is using to listen for OData connections.

Property name	Description
ODataSecureAddresses	A semicolon-delimited lists of TCP/IP address and ports that the OData server is using to listen for secure OData connections.
OmniIdentifier	Reserved for system use.
PacketsReceived	The number of client/server communication packets received. This value is not updated for HTTP or HTTPS connections.
PacketsReceivedUncomp	The number of packets that would have been received during client/server communications if compression was disabled. (This value is the same as the value for PacketsReceived if compression is disabled.)
PacketsSent	The number of client/server communication packets sent. This value is not updated for HTTP or HTTPS connections.
PacketsSentUncomp	The number of packets that would have been sent during client/server communications if compression was disabled. (This value is the same as the value for PacketsSent if compression is disabled.)
PageSize	The size of the database server cache pages. This can be set using the -gp option, otherwise, it is the maximum database page size of the databases specified on the command line.
ParameterizationPrepare-Count	The number of prepares for statements that have been automatically parameterized.
PeakCacheSize	The largest value the cache has reached in the current session, in kilobytes.
PlanStatisticsStored	For internal use only.
PlanStatisticsStoredMax	For internal use only.
Platform	The operating system on which the software is running.
PlatformVer	The operating system on which the software is running, including build numbers, service packs, and so on.
PrepStmt	The number of prepared statements currently being maintained by the database server for all databases and connections.
ProcessCPU	The CPU usage for the database server process. Values are in seconds. This property is supported on Windows, UNIX, and Linux systems. The value is cumulative since the database server was started. The value will not match the instantaneous value displayed in applications such as the Windows Task Manager or the Windows Performance Monitor. ProcessCPU, ProcessCPUSystem, and ProcessCPUUser have a fairly poor resolution on Windows. For example, if the resolution is 0.015625 seconds (1/64 second), it means you could execute some queries thousands of times before the value changes.

Property name	Description
ProcessCPUSystem	The CPU usage for the database server process CPU. This is the amount of CPU time that the database server spent inside the operating system kernel. Values are in seconds. This property is supported on Windows, UNIX, and Linux systems. The value is cumulative since the database server was started. The value will not match the instantaneous value displayed in applications such as the Windows Task Manager or the Performance Monitor. ProcessCPU, ProcessCPUSystem, and ProcessCPUUser have a fairly poor resolution on Windows. For example, if the resolution is 0.015625 seconds (1/64 second), it means you could execute some queries thousands of times before the value changes.
ProcessCPUUser	User CPU usage for the database server process. Values are in seconds. This excludes the amount of CPU time that the database server spent inside the operating system kernel. This property is supported on Windows, UNIX, and Linux systems. The value is cumulative since the database server was started. The value will not match the instantaneous value displayed in applications such as the Windows Task Manager or the Performance Monitor. ProcessCPU, ProcessCPUSystem, and ProcessCPUUser have a fairly poor resolution on Windows. For example, if the resolution is 0.015625 seconds (1/64 second), it means you could execute some queries thousands of times before the value changes.
ProcessID	The process ID of the database server process.
ProcessorAffinity	The logical processors being used by the database server as specified by the -gta option or by the sa_server_option system procedure and the ProcessorAffinity option.
ProcessorArchitecture	A string that identifies the processor type that the current software was built for. The following are possible values for the indicated operating system. **Windows:** X86, X86_64 **Linux:** X86, X86_64, ARM, ARM_64 **macOS:** X86_64 **Solaris:** SPARC, X86_64 **AIX:** PPC **HP-UX:** IA64 X86 represents a 32-bit database server. X86_64 represents a 64-bit database server.
ProductName	The name of the software.
ProductVersion	The version of the software being run.
ProfileFilterConn	The ID of the connection being monitored if procedure profiling for a specific connection is turned on. If profiling is not turned on, the value is an empty string.
ProfileFilterUser	The user ID being monitored if procedure profiling for a specific user is turned on. If procedure profiling for a specific user is not turned on, the value is an empty string.

Property name	Description
PropertyHistoryList	The current minimal set of properties being tracked.
PropertyHistoryListActual	The current set of properties being tracked.
PropertyHistorySize	Indicates either the minimum amount of time to store tracked property values or the maximum amount of memory to use to store tracked property values.
PropertyHistorySizeBytes	The current amount of memory, in bytes, that is currently being used for property history tracking.
QueryHeapPages	The number of cache pages used for query processing (hash and sort operations).
QueryMemActiveCurr	The number of requests actively using query memory.
QueryMemActiveEst	The database server's estimate of the steady state average of the number of requests actively using query memory.
QueryMemActiveMax	The maximum number of requests that are allowed to actively use query memory.
QueryMemExtraAvail	The number of pages available to grant beyond the base memory-intensive grant.
QueryMemGrantBase	The minimum number of pages granted to all requests.
QueryMemGrantBaseMI	The minimum number of pages granted to memory-intensive requests.
QueryMemGrantExtra	The number of query memory pages that can be distributed among active memory-intensive requests beyond QueryMemGrantBaseMI.
QueryMemGrantFailed	The total number of times a request waited for query memory, but failed to get it.
QueryMemGrantGranted	The number of pages currently granted to requests.
QueryMemGrantRequested	The total number of times any request attempted to acquire query memory.
QueryMemGrantWaited	The total number of times any request waited for query memory.
QueryMemGrantWaiting	The current number of requests waiting for query memory.
QueryMemPages	The number of pages available for query execution algorithms.
QueryMemPercentOfCache	The percentage of maximum cache size available for query execution algorithms.
QuittingTime	The shutdown time for the server. If none is specified, the value is none. If the database has the time_zone option set, then the value is returned using the database's time zone.
RememberLastPlan	Whether the database server is recording the last query optimization plan returned by the optimizer.
RememberLastStatement	Whether the database server is recording the last statement prepared by each connection.

Property name	Description
RemoteCapability	The remote capability name associated with a given capability ID.
RemoteputWait	The number of times the server had to block while sending a communication packet. Typically, blocking only occurs if the database server is sending data faster than the client or network can receive it. It does not indicate an error condition.
Req	The number of times the server has been asked to handle a new request or continue processing an existing request.
ReqCountActive	The number of active requests.
ReqCountBlockContention	The number of times that any connection has blocked due to contention for an internal server resource.
ReqCountBlockIO	The number of times that any connection has blocked while waiting for an IO request to complete.
ReqCountBlockLock	The number of times that any connection has blocked while waiting for a row lock held by another connection.
ReqCountUnscheduled	The number of times that any connection has blocked while waiting for a server thread to process it.
ReqTimeActive	The total amount of time in seconds that the server has spent directly servicing requests.
ReqTimeBlockContention	The total amount of time in seconds that any connection has blocked due to contention for an internal server resource.
ReqTimeBlockIO	The total amount of time in seconds that any connection has blocked while waiting for an IO request to complete.
ReqTimeBlockLock	The total amount of time in seconds that any connection has blocked while waiting for a row lock held by another connection.
ReqTimeUnscheduled	The total amount of time in seconds that any connection has blocked while waiting for a server thread to process it.
RequestFilterConn	The ID of the connection that logging information is being filtered for. If no filtering is being performed, the value is -1.
RequestFilterDB	The ID of the database that logging information is being filtered for. If no filtering is being performed, the value is -1.
RequestLogFile	The name of the request logging file, or an empty string if there is no request logging.
RequestLogging	The current setting for request logging. Values can be one of SQL, PLAN, HOSTVARS, PROCEDURES, TRIGGERS, OTHER, BLOCKS, REPLACE, ALL, or NONE.
RequestLogMaxSize	The maximum size of the request log file.

Property name	Description
RequestLogNumFiles	The number of request log files being kept.
RequestsReceived	The number of client/server communication requests or round trips. It is different from PacketsReceived in that multi-packet requests count as one request, and liveness packets are not included.
RequestTiming	Whether logging of request timing information is turned on. The logging of request timing information is turned on using the -zt database server option.
Rlbk	The number of rollback requests that have been handled.
SendFail	The number of times that the underlying communications protocols have failed to send a packet.
ServerEdition	A space-separated list of words describing the database server type. Values include: • Evaluation • Developer • Web • Educational • Standard • Advanced • Workgroup • OEM • Authenticated If you have a separate license for any of the following features, then the appropriate string(s) are added to the license string value: • HighAvailability • InMemory • FIPS
ServerName	The real server name (never an alternate server name). You can use this value to determine which of the operational servers is currently acting as primary in a database mirroring configuration.
SharedMemoryListener	Whether the database server is accepting shared memory connections, and No otherwise.
SingleCLR	The version number of the CLR if the database server uses a single CLR external environment for all databases or NONE if the database server uses one CLR external environment per database when executing CLR stored procedures.
SingleJVM	Whether the database server uses a single Java VM for all databases running on the database server (Yes), or whether the database server uses one Java VM per database when executing Java stored procedures (No).
StartDBPermission	The setting of the -gd server option, which can be one of DBA, all, or none.

Property name	Description
StartTime	The date/time that the server started. If the database has the time_zone option set, then the value is returned using the database's time zone.
StatisticsCleaner	Enable statistics cleaner to fix statistics that give bad estimates by performing scans on the table.
StreamsUsed	The number of database server streams in use.
TcpIpAddresses	A semicolon-delimited list of the IP addresses that the server is listening to for Command Sequence and TDS connections from clients. For example: `(::1):2638;127.0.0.1:2638`
TcpIpListeners	A semicolon-delimited list of IP addresses and IP address:port pairs that the database server is using to listen for TCP/IP connections.
TempDir	The directory in which temporary files are stored by the server.
ThreadDeadlocksAvoided	The number of times a thread deadlock error was detected but not reported to client applications. When the database server starts, the value of this property is 0. To avoid thread deadlock errors, the database server dynamically increases the multiprogramming level. If the multiprogramming level cannot be increased, a thread deadlock error is returned to the client application and the ThreadDeadlocksReported property is incremented. The ThreadDeadlocksAvoided property is always 0 on the personal server because dynamic tuning of the multiprogramming level is disabled by default.
ThreadDeadlocksReported	The number of times a thread deadlock error was reported to client applications. When the database server starts, the value of this property is 0.
TimeZoneAdjustment	The number of minutes that must be added to the Coordinated Universal Time (UTC) to display time local to the server.
TopologyAwareScheduling	Whether topology aware scheduling is in use.
TotalBuffers	The total number of network buffers.
UniqueClientAddresses	The number of unique client network addresses connected to a network server, excluding shared memory and local TCP/IP connections. This is the number of seats currently used for per-seat licensing.
UnschReq	The number of requests that are currently queued up waiting for an available server worker.
UserDefinedCounterRate01	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the change in the value of the counter over time.

Property name	Description
UserDefinedCounter-Rate02	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the change in the value of the counter over time.
UserDefinedCounter-Rate03	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the change in the value of the counter over time.
UserDefinedCounter-Rate04	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the change in the value of the counter over time.
UserDefinedCounter-Rate05	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the change in the value of the counter over time.
UserDefinedCounter-Raw01	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the absolute value of the counter.
UserDefinedCounter-Raw02	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the absolute value of the counter.
UserDefinedCounter-Raw03	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the absolute value of the counter.
UserDefinedCounter-Raw04	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the absolute value of the counter.
UserDefinedCounter-Raw05	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the absolute value of the counter.
WebClientLogFile	The name of the web service client log file.
WebClientLogging	Whether web service client information is being logged to a file.

Related Information

Database Server Property Tracking [page 933]
User-defined Properties [page 932]
EXTENDED_PROPERTY Function [System]
Database Server Startup Options [page 390]
PROPERTY Function [System]

sa_eng_properties System Procedure
sa_server_option System Procedure
List of Connection Properties [page 880]
List of Database Properties [page 917]

1.5.7.3 List of Database Properties

Database properties are available for each connection to a database. Use the DB_PROPERTY system function to retrieve the value for an individual database property, or the sa_db_properties system procedure to retrieve the values of all database properties.

Example

The following statement returns the page size of the current database:

```
SELECT DB_PROPERTY ( 'PageSize' );
```

Use the sa_db_properties system procedure to retrieve the values of all database properties:

```
CALL sa_db_properties;
```

Database Properties

Property name	Description
AccentSensitive	Whether the database is accent sensitive (On) or not (Off). The value is FRENCH if the database uses French sensitivity rules.
Alias	The database name.
AlternateMirrorServerName	The alternate mirror server name associated with the database if one was specified.
AlternateServerName	The alternate server name associated with the database if one was specified.
ApproximateCPUTime	An estimate of the amount of CPU time accumulated by the database, in seconds. The value may differ from the actual value by as much as 50 percent, although typical variations are in the 5 to 10 percent range. On multi-processor computers, each CPU (or hyperthread or core) accumulates time, so the sum of accumulated times for all connections may be greater than the elapsed time.

Property name	Description
ArbiterState	The state of the arbiter server. When the database you are connected to is not mirrored, this property is *NULL*. Otherwise this property is *connected* when the arbiter server is connected to the primary server, or *disconnected* when it is not connected to the primary server.
AuditingTypes	The types of auditing currently enabled.
Authenticated	Whether the database has been authenticated.
BackupInProgress	Whether the database is currently being backed up.
BlankPadding	Whether the database has blank padding enabled.
BytesReceived	The number of bytes received during client/server communications. This value is updated for HTTP and HTTPS connections.
BytesReceivedUncomp	The number of bytes that would have been received during client/server communications if compression was disabled. This value is the same as the value for BytesReceived if compression is disabled.
BytesSent	The number of bytes sent during client/server communications. This value is updated for HTTP and HTTPS connections.
BytesSentUncomp	The number of bytes that would have been sent during client/server communications if compression was disabled. This value is the same as the value for BytesSent if compression is disabled.
CacheHits	The number of database page lookups satisfied by finding the page in the cache.
CacheRead	The number of database pages that have been looked up in the cache.
CacheReadIndInt	The number of index internal-node pages that have been read from the cache.
CacheReadIndLeaf	The number of index leaf pages that have been read from the cache.
CacheReadTable	The number of table pages that have been read from the cache.
CacheReadWorkTable	The number of cache work table reads.
Capabilities	The capability bits enabled for the database. This property is primarily for use by Technical Support.
CarverHeapPages	The number of heap pages used for short-term purposes such as query optimization.
CaseSensitive	Whether the database is case sensitive (On) or not (Off).
CatalogCollation	The identifier for the collation used for the catalog. This property has extensions that you can specify when querying the property value with the DB_EXTENDED_PROPERTY function.

Property name	Description
CharSet	The CHAR character set of the database. This property has extensions that you can specify when querying the property value with the DB_EXTENDED_PROPERTY function.
CheckpointLogCommitToDisk	The number of checkpoint log commits to disk.
CheckpointLogPagesInUse	The number of checkpoint log pages in use.
CheckpointLogPagesRelocated	The number of relocated checkpoint log pages.
CheckpointLogPagesWritten	The number of checkpoint log pages that have been written.
CheckpointLogSavePreimage	The number of preimages of database pages that are being added to the checkpoint log.
CheckpointLogSize	The size of the checkpoint log, in pages.
CheckpointLogWrites	The number of writes to the checkpoint log.
CheckpointUrgency	The time that has elapsed since the last checkpoint, as a percentage of the checkpoint time setting of the database.
Checksum	Whether global checksums are enabled for the database (a checksum is created for each database page). Checksums are always present for critical pages.
Chkpt	The number of checkpoints that have been performed.
ChkptFlush	The number of ranges of adjacent pages written out during a checkpoint.
ChkptPage	The number of transaction log checkpoints.
CleanablePagesAdded	The number of pages marked to be cleaned since database server startup.
CleanablePagesCleaned	The number of database pages cleaned since database server startup.
CleanableRowsAdded	The number of rows marked to be deleted since database server startup.
CleanableRowsCleaned	The number of shadow table rows deleted since database server startup.
ClientStmtCacheHits	The number of prepares that were not required for this database because of the client statement cache. This is the number of additional prepares that would be required if client statement caching was disabled.
ClientStmtCacheMisses	The number of statements in the client statement cache for this database that were prepared again. This is the number of times a cached statement was considered for reuse, but could not be reused because of a schema change, a database option setting, or a DROP VARIABLE statement.
Collation	The collation used by the database. This property has extensions that you can specify when querying the property value.

Property name	Description
Commit	The number of times the server has forced a flush of the disk cache. On Microsoft Windows, the disk cache doesn't need to be flushed if unbuffered (direct) I/O is used.
CommitFile	The number of times the server has forced a flush of the disk cache. On Microsoft Windows, the disk cache doesn't need to be flushed if unbuffered (direct) I/O is used.
ConnCount	The number of connections to the database. The property value does not include connections used for internal operations, but it does include connections used for events and external environment support. To obtain an accurate count of the number of licensed connections in use, execute the following statement: `SELECT COUNT( * ) FROM sa_conn_info( ) WHERE number < 100000000;`
ConnectedTime	The total length of time, in seconds, that all connections have been connected to the database. The value is updated only when a request completes for a connection or when a connection disconnects. As a result, the value can lag behind for connections that are idle or busy executing for a long time in the database server. The value includes time accrued by any connection, including database events and background server connections (such as the database cleaner).
ConnPoolCachedCount	The number of connections disconnected by the application but cached for connection pooling.
ConnPoolHits	The number of reused pooled connections.
ConnPoolMisses	The number of pooled connections which were cached but could not be reused.
ConnsDisabled	Whether connections to the current database are disabled (On) or not (Off).
CopyNodeParent	The name of the current parent server of a copy node in a read-only scale-out configuration.
CurrentRedoPos	The current offset in the transaction log file where the next database operation is to be logged.
CurrentTimelineID	For internal use only.
CurrentTimelineSignature	For internal use only.
CurrentTimezoneOffset	The number of minutes that the database is ahead of Coordinated Universal Time (UTC).
CurrIO	The current number of file I/Os that were issued by the server but haven't yet completed.
CurrRead	The current number of file reads that were issued by the server, but haven't yet completed.
CurrWrite	The current number of file writes that were issued by the server, but haven't yet completed.

Property name	Description
Cursor	The number of declared cursors that are currently being maintained by the database server.
CursorOpen	The number of open cursors that are currently being maintained by the database server.
DatabaseCleaner	Whether the database cleaner is enabled.
DBFileFragments	The number of database file fragments. This property is supported on Microsoft Windows.
DiskRead	The number of pages that have been read from disk.
DiskReadHint	The number of disk read hints.
DiskReadHintPages	The number of disk read hint pages.
DiskReadIndInt	The number of index internal-node pages that have been read from disk.
DiskReadIndLeaf	The number of index leaf pages that have been read from disk.
DiskReadTable	The number of table pages that have been read from disk.
DiskReadWorkTable	The number of disk work table reads.
DiskRetryReadScatter	The number of disk read retries for scattered reads.
DiskSandbox	Whether the read-write file operations of the database are restricted to the directory where the main database file is located.
DiskSyncRead	The number of disk reads issued synchronously.
DiskSyncWrite	The number of writes issued synchronously.
DiskWaitRead	The number of times the database server waited for an asynchronous read.
DiskWaitWrite	The number of times the database server waited for an asynchronous write.
DiskWrite	The number of modified pages that have been written to disk.
DiskWriteHint	The number of disk write hints.
DiskWriteHintPages	The number of disk gather write hints.
DriveBus	The type of bus to which the database file is connected: one of Unknown, SCSI, ATAPI, ATA, 1394, SSA, Fibre, USB, RAID, iSCSI, SAS, SATA, SD, MMC, Virtual, FileBackedVirtual. If the bus type cannot be determined, this property is NULL. The value of this property for the primary dbspace is recorded in the SYSHISTORY view.

Property name	Description
DriveModel	The model number of the drive on which the database is located. The value of this property for the primary dbspace is recorded in the SYSHISTORY view. If the drive model cannot be determined, this property is NULL.
DriveType	The type of drive on which the database file is located: one of CD, FIXED, RAMDISK, REMOTE, REMOVABLE, UNKNOWN. This property has extensions that you can specify when querying the property value with the DB_EXTENDED_PROPERTY function. Depending on the version of UNIX or Linux and the type of drive, it may not be possible to determine the drive type. In these cases the value is UNKNOWN.
Encryption	The type of encryption used for database or table encryption: one of None, Simple, AES, AES256, AES256CTR, AES_FIPS, AES256_FIPS, AES256CTR_FIPS, ARIA256 or ARIA256CTR. **i Note** The following algorithms are supported in database encryption only: AES256CTR, AES256CTR_FIPS, ARIA256, and ARIA256CTR.
EncryptionScope	The part of the database, if any, that can be encrypted. TABLE indicates that table encryption is enabled. DATABASE indicates that the whole database is encrypted. NONE indicates that table encryption is not enabled, and the database is not encrypted.
ExprCacheAbandons	The number of times that the expression cache was completely abandoned because the hit rate was too low.
ExprCacheDropsToReadOnly	The number of times that the expression cache dropped to read-only status because the hit rate was low.
ExprCacheEvicts	The number of evictions from the expression cache.
ExprCacheHits	The number of hits in the expression cache.
ExprCacheInserts	The number of values inserted into the expression cache.
ExprCacheLookups	The number of lookups performed in the expression cache.
ExprCacheResumesOfReadWrite	The number of times that the expression cache resumed read-write status because the hit rate increased.
ExprCacheStarts	The number of times the expression cache was started.
ExtendDB	The number of pages by which the database file has been extended.
ExtendTempWrite	The number of pages by which temporary files have been extended.
File	The file name of the database root file, including path. This property has extensions that you can specify when querying the property value with the DB_EXTENDED_PROPERTY function.

Property name	Description
FileSize	The file size of the system dbspace, in pages. This property has extensions that you can specify when querying the property value with the DB_EXTENDED_PROPERTY function.
FreePages	The number of free pages in the system dbspace. The FreePages property is only supported on databases created with version 8.0.0 or later. This property has extensions that you can specify when querying the property value with the DB_EXTENDED_PROPERTY function.
FullCompare	The number of comparisons that have been performed beyond the hash value in an index.
GetData	The number of GETDATA requests.
GlobalDBID	The value of the global_database_id option used to generate unique primary key values in a replication environment.
HasCollationTailoring	Whether collation tailoring was specified when the database was created. Possible values are On or Off.
HasEndianSwapFix	Whether the database supports both big-endian and little endian UTF-16 encoding on all platforms, regardless of the endianness of the platform. Possible values are On or Off.
HashForcedPartitions	The number of times that a hash operator was forced to partition because of competition for memory.
HashRowsFiltered	The number of probe rows rejected by bit-vector filters.
HashRowsPartitioned	The number of rows written to hash work tables.
HashWorkTables	The number of work tables created for hash-based operations.
HasNCHARLegacyCollationFix	Whether the database was created by using version 11 or later software, or by using a version 10 database server with the legacy collation fix and that uses a legacy NCHAR collation (On). The value is Off for databases created using a version 10 database server without the legacy collation fix, or databases created using a version 10 database server that do not use a legacy NCHAR collation.
HasQDADTTFeature	Whether the queue depth checking feature is available (On) or not (Off).
HasTornWriteFix	Whether the database has a fixed file format to allow recovery from partial writes.
HeapsCarver	The number of heaps used for short-term purposes such as query optimization.
HeapsLocked	The number of relocatable heaps currently locked in the cache.
HeapsQuery	The number of heaps used for query processing (hash and sort operations).
HeapsRelocatable	The number of relocatable heaps.

Property name	Description
HttpClientMultipleHeaders	Whether the database supports multiple headers with the same name in an HTTP result set.
HttpConnPoolCachedCount	The absolute count of cached database connections within all pools.
HttpConnPoolHits	The rate of connections reused by the same service.
HttpConnPoolMisses	The rate of new connections that were allocated when a connection could not be accessed from a pool. This property only counts HTTP requests to services that use connection pooling. At the time of access, a connection may not have been available due to a small pool size whose oldest connection did not fit the steal criteria.
HttpConnPoolSteals	The rate of connections taken by services where the connection belonged to another service. The service steals a connection from another service if the criteria is met for an HTTP request for a service not having any direct reusable connections and the pool size and age of the least used connection. A connection is allocated for the service and HttpConnPoolMisses is incremented instead if the pool criteria is not met.
IdentitySignature	This property is for internal use only.
IdentitySignatureUUID	This property is for internal use only.
IdleCheck	The number of times that the server's idle thread has become active to do idle writes, idle checkpoints, and so on.
IdleChkpt	The number of checkpoints completed by the server's idle thread. An idle checkpoint occurs whenever the idle thread writes out the last dirty page in the cache.
IdleChkTime	The number in hundredths of a second spent checkpointing during idle I/O.
IdleWrite	The number of disk writes that have been issued by the server's idle thread.
IndAdd	The number of entries that have been added to indexes.
IndLookup	The number of entries that have been looked up in indexes.
IOParallelism	The estimated number of simultaneous I/O operations supported by the dbspace. This property has extensions that you can specify when querying the property value with the DB_EXTENDED_PROPERTY function.
IOToRecover	The estimated number of I/O operations required to recover the database.
IQStore	This property is for internal use only.
JavaVM	The Java VM the database server uses to execute Java in the database.
KeyDerivationIterations	The number of times that the database encryption key was hashed when the database was created.

Property name	Description
Language	A comma-separated list of languages known to be supported by the database collation. The languages are in two-letter ISO format. If the language isn't known, the return value is NULL.
LastCheckpointTime	The date and time, in milliseconds, of the most recent checkpoint. If the database has the time_zone option set, then the value is returned using the database's time zone.
LastCommitRedoPos	The redo log position after the last COMMIT operation was written to the transaction log by the database.
LastSyncedRedoPos	The last redo position for which a write was issued to disk and the data was synchronized to the physical medium. Data prior to this position is expected to be present on disk even in the event of a power failure.
LastWrittenRedoPos	The last redo position for which a write was issued to disk. The write is not necessarily synchronized to the physical medium as it may still be cached by the operating system, disk controller, or disk drive.
LockCount	The number of locks held by the database.
LockTablePages	The number of pages used to store lock information.
LogFileFragments	The number of log file fragments. This property is supported on Microsoft Windows.
LogFreeCommit	The number of redo free commits. A redo free commit occurs when a commit of the transaction log is requested but the log has already been written (so the commit was done for free).
LogMirrorName	The file name of the transaction log mirror, including the path.
LogName	The file name of the transaction log, including path.
LogWrite	The number of pages that have been written to the transaction log.
LTMGeneration	This property is for internal use only.
LTMTrunc	This property is for internal use only.
MaxConnections	The maximum number of concurrent connections that are allowed to the database. This property corresponds to the max_connections database option.
MaxIO	The maximum value that CurrIO has reached.
MaxRead	The maximum value that CurrRead has reached.
MaxWrite	The maximum value that CurrWrite has reached.
MirrorMode	Whether database mirroring is not in use (NULL), or if the mirroring mode specified with the -xp option is synchronous or asynchronous.

Property name	Description
MirrorRole	The role of the database server in a mirroring configuration. The value is Primary when the database is the primary database server or the mirror server. The value is Mirror when the database server is the mirror server. The value is Copy when the database server is a read-only mirror server that obtains its log pages from the current primary server, the current mirror server, or another read-only node.
MirrorServerState	The state of the mirror server. This property has extensions that you can specify when querying the property value with the DB_EXTENDED_PROPERTY function.
MirrorServerWaits	The number of times the database server waited more than 500 milliseconds when sending log pages to copy servers.
MirrorState	Information about the database's status in a mirroring configuration. This property has extensions that you can specify when querying the property value with the DB_EXTENDED_PROPERTY function.
MultiByteCharSet	Whether the database uses a multibyte character set, such as UTF-8, (On), or not (Off).
Name	The database name (identical to Alias).
NcharCharSet	The NCHAR character set of the database.
NcharCollation	The name of the collation used for NCHAR data. This property has extensions that you can specify when querying the property value.
NextScheduleTime	The next scheduled execution time for a specified event.This property has extensions that you can specify when querying the property value.
OptionWatchAction	The action that is taken when an attempt is made to set a database option that is included in the OptionWatchList property.
OptionWatchList	The list of database options being monitored by the database server.
PacketsReceived	The number of client/server communication packets received. This value is not updated for HTTP or HTTPS connections.
PacketsReceivedUncomp	The number of packets that would have been received during client/server communications if compression was disabled. (This value is the same as the value for PacketsReceived if compression is disabled.)
PacketsSent	The number of client/server communication packets sent. This value is not updated for HTTP or HTTPS connections.
PacketsSentUncomp	The number of packets that would have been sent during client/server communications if compression was disabled. (This value is the same as the value for PacketsSent if compression is disabled.)
PageRelocations	The number of relocatable heap pages that have been read from the temporary file.

Property name	Description
PageSize	The page size of the database, in bytes.
ParameterizationPrepareCount	The number of prepares for statements that have been automatically parameterized.
PartnerState	Information about the database server's status in a mirroring configuration. Values include: *Connected*, when there is a connection from this server to a partner server and a connection from the partner server to this server; *Disconnected* when there are no connections between this server and the partner server; Incoming only when there is a there is a connection from the partner server to this server; and *Outgoing* only when there is a connection from this server to the partner server. The value is *NULL* when the database server is not involved in a mirroring configuration.
PlanStatisticsStored	For internal use only.
Prepares	The number of statement prepares performed for the database.
PrepStmt	The number of prepared statements currently being maintained by the database server for the current database.
PreviousTimelineID	For internal use only.
ProcedurePages	The number of relocatable heap pages that have been used for procedures.
ProcedureProfiling	Whether procedure profiling is turned on for the database.
PropertyHistoryList	The current set of properties being tracked for this database.
QueryBypassed	The number of requests reused from the plan cache.
QueryBypassedCosted	The number of requests processed by the optimizer bypass using costing.
QueryBypassedHeuristic	The number of requests processed by the optimizer bypass using heuristics.
QueryBypassedOptimized	The number of requests initially processed by the optimizer bypass and subsequently fully optimized by the optimizer.
QueryCachedPlans	The number of cached execution plans across all connections.
QueryCachePages	The number of pages used to cache execution plans.
QueryDescribedBypass	The number of describe requests processed by the optimizer bypass.
QueryDescribedOptimizer	The number of describe requests processed by the optimizer.
QueryHeapPages	The number of cache pages used for query processing (hash and sort operations).
QueryJHToJNLOptUsed	The number of times a hash join was converted to a nested loops join.

Property name	Description
QueryLowMemoryStrategy	The number of times the server changed its execution plan during execution as a result of low memory conditions. The strategy can change because less memory is available than the optimizer estimated, or because the execution plan required more memory than the optimizer estimated.
QueryMemActiveCurr	The number of requests actively using query memory.
QueryMemGrantFailed	The total number of times a request waited for query memory, but failed to get it.
QueryMemGrantGranted	The number of pages currently granted to requests.
QueryMemGrantRequested	The total number of times any request attempted to acquire query memory.
QueryMemGrantWaited	The total number of times any request waited for query memory.
QueryMemGrantWaiting	The current number of requests waiting for query memory.
QueryOpened	The number of OPEN requests for execution.
QueryOptimized	The number of requests fully optimized.
QueryReused	The number of reused query plans.
QueryRowsFetched	The number of rows that have been read from base tables, either by a sequential scan or an index scan, for this database.
QueryRowsMaterialized	The number of rows written to work tables during query processing.
ReadOnly	Whether the database is being run in read-only mode (On) or not (Off).
ReceivingTracingFrom	The name of the database from which the tracing data is coming. The value is a blank string if tracing is not attached.
RecoveryUrgency	An estimate of the amount of time required to recover the database as a percentage of the recovery time setting of the database.
RecursiveIterations	The number of iterations for recursive unions.
RecursiveIterationsHash	The number of times recursive hash join used a hash strategy.
RecursiveIterationsNested	The number of times recursive hash join used a nested loops strategy.
RecursiveJNLMisses	The number of index probe cache misses for recursive hash join.
RecursiveJNLProbes	The number of times recursive hash join attempted an index probe.
RelocatableHeapPages	The number of pages used for relocatable heaps (cursors, statements, procedures, triggers, views, and so on.).
RemoteTrunc	The minimal confirmed log offset for the SQL Remote Message Agent.

Property name	Description
ReqCountActive	The number of requests processed, or NULL if the RequestTiming server property is set to Off.
ReqCountBlockContention	The number of times connections to the database waited for atomic access, or NULL if the -zt option was not specified.
ReqCountBlockIO	The number of times connections to the database waited for I/O to complete, or NULL if the -zt option was not specified.
ReqCountBlockLock	The number of times connections to the database waited for a lock, or NULL if the -zt option was not specified.
ReqCountUnscheduled	The number of times connections to the database waited for scheduling, or NULL if the -zt option was not specified.
ReqTimeActive	The amount of time spent in seconds processing requests, or NULL if the -zt option was not specified.
ReqTimeBlockContention	The amount of time spent in seconds waiting for atomic access, or NULL if the RequestTiming server property is set to Off.
ReqTimeBlockIO	The amount of time spent in seconds waiting for I/O to complete, or NULL if the -zt option was not specified.
ReqTimeBlockLock	The amount of time spent in seconds waiting for a lock, or NULL if the -zt option was not specified.
ReqTimeUnscheduled	The amount of unscheduled time, or NULL if the -zt option was not specified.
RequestsReceived	The number of client/server communication requests or round trips. This property is different from PacketsReceived in that multi-packet requests count as one request, and liveness packets are not included.
Rlbk	The number of rollback requests that have been handled.
RollbackLogPages	The number of pages in the rollback log.
SendingTracingTo	The connection string where the tracing data is being sent. The value is a blank string if tracing is not attached.
SnapshotCount	The number of snapshots associated with the database.
SnapshotIsolationState	Whether snapshot isolation is enabled for the database (On) or not (Off), or whether it will be enabled (in_transition_to_on) or disabled (in_transition_to_off) once the current transactions complete.
SortMergePasses	The number of merge passes used during sorting.
SortRowsMaterialized	The number of rows written to sort work tables.

Property name	Description
SortRunsWritten	The number of sorted runs written during sorting.
SortSortedRuns	The number of sorted runs created during run formation.
SortWorkTables	The number of work tables created for sorting.
StatementDescribes	The total number of statements processed by DESCRIBE requests.
StatementPostAnnotates	The number of statements processed by the semantic query transformation phase.
StatementPostAnnotatesSimple	The number of statements processed by the semantic query transformation phase, but which skipped some of the semantic transformations.
StatementPostAnnotatesSkipped	The number of statements that have completely skipped the semantic query transformation phase.
SynchronizationSchemaChangeActive	Whether an active connection executed a START SYNCHRONIZATION SCHEMA CHANGE statement but has not executed a STOP SYNCHRONIZATION SCHEMA CHANGE statement (On) or not (Off).
SyncTrunc	The minimal confirmed log offset for the MobiLink client dbmlsync executable.
TempFileName	The file name of the database temporary file, including path.
TempTablePages	The number of pages in the temporary file used for temporary tables.
TimelineBranchOffset	For internal use only.
TimeWithoutClientConnection	The time in seconds since the last command sequence or TDS client connection, or the time since the database started if no connections have been made to the database. The value is 0 if there is a current connection.
TimeZone	The time zone that the database uses for time zone calculations. The value is NULL if the Timezone database option is not set.
TriggerPages	The number of relocatable heap pages used for triggers.
UserDefinedCounterRate01	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the change in the value of the counter over time.
UserDefinedCounterRate02	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the change in the value of the counter over time.

Property name	Description
UserDefinedCounterRate03	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the change in the value of the counter over time.
UserDefinedCounterRate04	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the change in the value of the counter over time.
UserDefinedCounterRate05	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the change in the value of the counter over time.
UserDefinedCounterRaw01	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the absolute value of the counter.
UserDefinedCounterRaw02	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the absolute value of the counter.
UserDefinedCounterRaw03	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the absolute value of the counter.
UserDefinedCounterRaw04	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the absolute value of the counter.
UserDefinedCounterRaw05	The current value of the user-defined performance counter. The semantics of this property are defined by the client application. This counter can also be accessed from the Performance Monitor. The Performance Monitor displays the absolute value of the counter.
UTCTimestampCatalog	Whether the database is storing UTC timestamps in the system tables (On) or not (Off).
VersionStorePages	The number of pages in the temporary file that are being used for the row version store when snapshot isolation is enabled.
ViewPages	The number of relocatable heap pages used for views.

Property name	Description
WriteChecksum	Whether the database server adds checksums to pages before they are written out (On) or not (Off).
XPathCompiles	The number of times any XPath query (using the OPENXML operator) was compiled by the database server since database server startup.

Related Information

Database Options [page 651]
DB_PROPERTY Function [System]
DB_EXTENDED_PROPERTY Function [System]
sa_db_properties System Procedure
sa_db_option System Procedure
List of Database Server Properties [page 900]
List of Connection Properties [page 880]

1.5.7.4 User-defined Properties

User-defined properties are available for each connection to a database.

A set of user-defined properties is supported. Your application must define the semantics for each user-defined property and set the values that are returned. The property values can be set to an absolute 32-bit UNSIGNED INTEGER value using the sa_user_defined_counter_set system procedure or incremented using the sa_user_defined_counter_add system procedure. The sa_user_defined_counter_add system procedure can also be used to decrement the property by adding a negative value. The user-defined counters are:

- UserDefinedCounterRaw01
- UserDefinedCounterRaw02
- UserDefinedCounterRaw03
- UserDefinedCounterRaw04
- UserDefinedCounterRaw05
- UserDefinedCounterRate01
- UserDefinedCounterRate02
- UserDefinedCounterRate03
- UserDefinedCounterRate04
- UserDefinedCounterRate05

The values for user-defined properties are maintained separately for each connection, database, and server. The current value of the properties can be retrieved using the CONNECTION_PROPERTY, DB_PROPERTY, and PROPERTY functions, respectively.

One of example of using these properties is to have a raw property that shows the current number of keys remaining in a key pool. The value of the property would fluctuate up and down over time. At the same time, a rate property could show how quickly keys were being used from the key pool.

Related Information

SQL Functions Used to Monitor Statistics [page 1504]
Windows Performance Monitor [page 1506]
sa_user_defined_counter_add System Procedure
sa_user_defined_counter_set System Procedure

1.5.7.5 Database Server Property Tracking

The database server can store the values of numeric database server properties so that you can track the changes of numeric database server properties over time.

Tracking database server property values over time aids in evaluating the overall health of a database server. For example, a brief increase in CPU usage may not be an issue, but if the CPU usage is at 100 percent for an extended period of time, it could indicate that the hardware is insufficient.

Rather than having to poll, analyze, and store database server property values at regular intervals, configure the database server to track database server properties that return numeric values. These values are stored in memory for a period of time so that polling can be done at larger intervals, reducing the load on the database server.

When database server property tracking is enabled, property values are tracked at fixed intervals and you can query historic property values by using the sp_property_history system procedure.

In this section:

Viewing All Trackable Database Server Property Values [page 934]
View the list of database server property values that can be tracked.

Configuring Database Server Property Tracking [page 935]
Configure your database server to track the values of numeric database server properties.

Related Information

sp_property_history System Procedure
PROPERTY_IS_TRACKABLE Function [System]
-sf Database Server Option [page 493]
sa_server_option System Procedure
sa_db_option System Procedure
-phl Database Server Option [page 480]

-phs Database Server Option [page 481]
List of Database Server Properties [page 900]
List of Database Properties [page 917]
System Privileges [page 1577]

1.5.7.5.1 Viewing All Trackable Database Server Property Values

View the list of database server property values that can be tracked.

Context

Only database server properties with numeric values can be tracked.

You can find the PropNum of the database server property by running the sa_eng_properties system procedure or by calling the PROPERTY_NUMBER function.

Procedure

Execute the following statement:

```
SELECT * FROM sa_eng_properties()
WHERE PROPERTY_IS_TRACKABLE( PropNum ) = 1;
```

Results

The list of database server properties that can be tracked is returned.

Related Information

sp_property_history System Procedure
PROPERTY_IS_TRACKABLE Function [System]
-sf Database Server Option [page 493]
sa_server_option System Procedure
sa_db_option System Procedure
-phl Database Server Option [page 480]
-phs Database Server Option [page 481]

List of Database Server Properties [page 900]
List of Database Properties [page 917]
System Privileges [page 1577]
PROPERTY_NUMBER Function [System]
sa_eng_properties System Procedure

1.5.7.5.2 Configuring Database Server Property Tracking

Configure your database server to track the values of numeric database server properties.

Context

Only database server properties with numeric values can be tracked.

Procedure

Choose one or both of the following options:

Option	Action
Specify database server properties to be tracked for the database server	• When starting the database server, use the -phl and -phs database server options to turn on history tracking for a list of specified database server properties and to specify the maximum amount of memory to use for tracking property history. For example, run the following command: *dbsrv17* `-n myserver -phl ProcessCPUSystem, ProcessCPUUser -phs 250K` • If the database server is already running, then use the sa_server_option system procedure to configure property tracking for the database server. For example, execute the following statement: `CALL sa_server_option( 'PropertyHistoryList,ProcessCPUSystem,ProcessCPUUser,PropertyHistorySize,250K' );`
Specify database server properties to be tracked for the database	Use the sa_db_option system procedure to configure property tracking of database server properties for the database. For example, execute the following statement: `CALL sa_db_option( 'PropertyHistoryList,ProcessCPUSystem,ProcessCPUUser' );`

Results

The values of the specified database server properties are tracked for either the specified amount of time or until the specified maximum amount of memory has been reached.

Related Information

sp_property_history System Procedure
PROPERTY_IS_TRACKABLE Function [System]
-sf Database Server Option [page 493]
sa_server_option System Procedure
sa_db_option System Procedure
-phl Database Server Option [page 480]
-phs Database Server Option [page 481]
List of Database Server Properties [page 900]
List of Database Properties [page 917]
System Privileges [page 1577]

1.5.8 Physical Limitations on Size and Number of Databases

This topic lists the physical limitations on size and number of objects in a SQL Anywhere database. Typically, the memory, CPU, and disk drive of the computer are the most limiting factors.

Item	Limitation
Database size	13 files per database. For each file, the largest file allowed by operating system and file system
Dbspace size	2^{28} x page size
Temporary file size	2^{28} x page size
Field size	2 GB
File size (FAT 12)	16 MB
File size (FAT 16)	2 GB
File size (FAT 32)	4 GB
File size for NTFS, HP-UX 11.0 and later, Solaris 2.6 and later, Linux 2.4 and later, AIX, macOS	• 512 GB for 2 KB pages • 1 TB for 4 KB pages • 2 TB for 8 KB pages
File size (all other platforms and file systems)	2 GB

Item	Limitation
Maximum cache size (Microsoft Windows Server 2003 Web Edition, Microsoft Windows Server 2003 Standard Edition, Microsoft Windows Server 2008, Microsoft Windows Server 2008 R2, Microsoft Windows 7)	1.8 GB
Maximum cache size (Microsoft Windows Server 2003 Enterprise Edition, Microsoft Windows Server 2003 Datacenter Edition)	2.7 GB
Maximum cache size (UNIX/Linux: Solaris, x86 Linux, IBM AIX, HP-UX)	2 GB for 32-bit servers
Maximum cache size (Win 64)	Limited by physical memory on 64-bit servers
Maximum cache size (Itanium HP-UX)	Limited by physical memory on 64-bit servers
Maximum index entry size	No limit
Number of databases per server	255
Number of columns per table	45000 An excessive number of columns, although allowed, can affect performance.
Number of nullable columns per table	min(45000, (page size - overhead) * 8)
Number of columns in a procedure result set	45000
Number of columns in a SELECT list	100000
Number of columns in a GROUP BY list	100000
Number of columns in a GROUP BY with grouping sets	64
Number of columns in a CUBE	15
Number of distinct grouping sets	32768
Length of DEFAULT for a column	32768
Length of COMPUTE for a column	32768
Length of DEFAULT for procedure parameters	32768
Length of DEFAULT for a user-defined domain	32768
Length of check constraints	2 GB
Number of indexes per table	2^{32}
Number of rows per database	4096 x 2^{28} x 13
Number of rows per table	4096 x 2^{28}
Number of base and global temporary tables per database	2^{32} - 2^{20} - 1 = 4293918719
Number of local temporary tables per connection	2^{20} = 1048576
Number of tables referenced per transaction	No limit
Number of stored procedures per database	2^{32} - 1 = 4294967295
Number of concurrent statements per database server	20 x `number-of-database-connections` + 65534
Number of events per database	2^{31} - 1 = 2147483647

Item	Limitation
Number of triggers per database	$2^{32} - 1 = 4294967295$
Row size	Limited by file size
Table size	Maximum file size. User-created indexes for the table can be stored separately from the table
Character data types	$2^{31} - 1 = 2147483647$
Binary data types	$2^{31} - 1 = 2147483647$
Row size	45000 fields
Array size	6.4 million elements
Identifiers (including user IDs, table names, and column names)	128 bytes
Passwords	255 bytes
Database server names	250 bytes (TCP/IP and shared memory
Database names	250 bytes
Number of secure feature keys per database server	1000

In this section:

SQL Anywhere Hardware Requirements [page 938]
SQL Anywhere has several hardware requirements.

Related Information

List of Database Server Properties [page 900]

1.5.8.1 SQL Anywhere Hardware Requirements

SQL Anywhere has several hardware requirements.

For information about hardware requirements, see http://scn.sap.com/docs/DOC-35653

Related Information

SAP SQL Anywhere Hardware Requirements

1.6 Database Maintenance

Several tools and methods are provided for maintaining a database.

In this section:

1.6.1 Database Backup and Recovery

Use backups to restore all committed changes to a database up to the time it became unavailable.

A **backup** is a full or partial copy of the information in a database, held in a physically separate location. If the main database file becomes unavailable, you can **restore** a copy of it from the backup, and use the current transaction log to **recover** any additional data that may not be present in the database you restored.

There are several types of backup, and several ways of restoring and recovering data.

Two very important best practices for backup and recovery is 1) to adhere to a suitable backup strategy that includes validating that you can restore from your backups, and 2) to restore from *copies* of your backup files, not the backup files themselves.

In this section:

1.6.1.1 Database Backup

Backing up a running database provides a snapshot of the database where the data is in a consistent state, even though other users are modifying the database.

A backup includes all completed transactions in the transaction log before the final page of the transaction log is read, and possibly additional operations started after the backup was initiated, depending on the options chosen for the backup.

At the start of a backup, the database server issues a checkpoint. At the end of a backup, the database server flags the backup copy of the database as needing recovery. This step causes any operations that happened since the start of the backup to be applied when the backup copy of the database is started. It also causes operations that were incomplete at the checkpoint to be undone if they were not committed.

The database server prevents the following operations from being executed while a backup is in progress:

- Another backup, with the exception of a live backup.
- A checkpoint, other than one issued by the backup instruction.
- Any statement that causes a checkpoint.

Supported backup tools and methods

There are several backup tools and methods you can use to perform backups:

- BACKUP DATABASE statement
- Backup utility (dbbackup)
- *Create Backup Images Wizard*
- *Backup Database Wizard*

You can also use the DBBackup method of the database tools interface (DBTools), although this method is not described here. Additionally, support for the Microsoft Volume Shadow Copy Service (VSS) is provided. Microsoft VSS can be used for creating snapshots of files as backups.

Database mirroring (high availability), read-only scale-out, and transaction log mirroring are also approaches that help you ensure continued access to data in the event of a media failure, though they are not considered backup methods.

In this section:

When you make a backup, unless you specify otherwise, the database server makes a copy of the transaction log and leaves the transaction log in place to continue growing.

Considerations for Databases Involved in Synchronization and Replication [page 959]
For synchronization and replication environments, use backup options to rename and restart the transaction log.

Related Information

SQL Anywhere Volume Shadow Copy Service (VSS) [page 945]
Transaction Log Mirrors [page 301]
High Availability and Read-only Scale-out Systems [page 1756]
Database Recovery [page 960]
Backup Utility (dbbackup) [page 1121]
BACKUP DATABASE Statement

1.6.1.1.1 Types of Backup

There are many choices to make when deciding when, where, and how to perform a backup. The decisions you make define the backup type to perform (for example, server-side and client-side, full and incremental).

Backup types are not mutually exclusive. For example, you may decide to perform a full, online, server-side, image backup. This means you perform a full backup of a running database and its transaction log, initiating the backup from the database server computer, and store the copies of the database and transaction log as separate files on the database server computer.

> → Tip
>
> A popular and efficient backup strategy is to perform server-side, full, image backups to a separate device regularly, with more frequent incremental backups between full backups. Remember to test recovering from a backup to ensure that in the event of an actual failure, recovery would be successful.

The following table summarizes the types of backups you can perform, and when applicable, the tools you can use to perform them. Many types have a related alternative. These types are defined alongside their alternatives.

Backup terminology	Explanation	Performed using:
Server-side and client-side backups	A **server-side backup** is performed by the database server. This type of backup is generally faster than client-side, because data does not need to be transmitted to another computer. A **client-side backup** is a backup that is performed by the Backup utility (dbbackup). However, there is one exception to this. Running dbbackup with the -s option tells the database server to perform the backup using a BACKUP DATABASE statement instead, making it a server-side backup.	You perform a server-side backup using any of the backup tools provided. When using the Backup utility (dbbackup), you must specify the -s option. You perform a client-side backup using the Backup utility (dbbackup).
Full and incremental backups	A **full backup** consists of a copy of the database files and the transaction log. Full backups are beneficial because they include all your data, but they can take time, especially if your database is large. A common backup approach is to perform periodic full backups (for example, weekly), interspersed with more frequent incremental backups (for example, daily). An **incremental backup** consists of a copy of the transaction log only. Run a full backup before running an incremental back up. Incremental backups (interspersed with full backups) are the most common backup. They take much less time than full backups because you only copy the transaction log.	You perform a full or incremental backup using any of the backup tools provided, with the exception of an incremental. However, you cannot perform an incremental archive backup using the *Backup Database Wizard* in SQL Central.
Image and archive backups	An **image backup** copies the database file and/or the transaction log, and stores each file separately. An image backup is easier to restore from than archive backups. This is the most commonly chosen backup format.	You perform an image backup using: • BACKUP DATABASE statement • Backup utility (dbbackup) • *Create Backup Images Wizard* in SQL Central (image backups) • The Cockpit

Backup terminology	Explanation	Performed using:
Online and offline backups	An **online backup** is performed against a running database. Backing up a running database provides a snapshot of the database where the data is in a consistent state, even though other users are modifying the database. Typically, backups are performed as online backups. An **offline backup** is a copy of the database files made when the database is not running. An offline backup copy must only be performed when the database has been stopped normally, and the database server has been shut down properly.	You perform a live backup using any of the backup tools provided. You perform an offline backup using tools such as third-party backup software, or by using the file management features provided by your operating system.
Live or continuous backups	A **live backup** (also known as a **continuous backup**) is a backup of the transaction log that runs continuously while the database is running.	You perform a live backup using the Backup utility (dbbackup) and specifying the -l option.
Parallel backup	A **parallel backup** is a server-side backup that makes use of device-level parallelism to decrease the overall time required to complete a backup operation.	You perform a parallel backup as part of a server-side backup, using either the BACKUP DATABASE statement or the Backup utility (dbbackup).

In this section:

Live Backups (Continuous Backups) [page 944]
A **live backup** is a continuous backup of the database that runs on a secondary computer.

SQL Anywhere Volume Shadow Copy Service (VSS) [page 945]
SQL Anywhere is compatible with the Microsoft Volume Shadow Copy Service (VSS).

Parallel Database Backups [page 945]
Parallel backups use physical device-level parallelism to decrease the overall time required to complete a backup operation.

Related Information

Backing up a Database Using the Backup Utility (dbbackup) [page 946]
Backing up a Database Using the BACKUP DATABASE Statement [page 949]
Backing up a Database Using SQL Central [page 951]

1.6.1.1.1.1 Live Backups (Continuous Backups)

A **live backup** is a continuous backup of the database that runs on a secondary computer.

In the event of media failure, your database can be restarted on the secondary computer. Depending on the load and performance on the computer the database server is running on, a live backup may not contain all committed transactions that occurred prior to the media failure.

Live backups are performed using the Backup utility (dbbackup), by specifying the -l option. Always run a live backup from a separate computer.

The live backup of the transaction log is always the same length as, or shorter than, the active transaction log. When a live backup is running, and another backup restarts the transaction log (dbbackup -r or dbbackup -x), the live backup automatically truncates the live backup log and restarts the live backup at the beginning of the new transaction log.

Live Backups Compared to Transaction Log Mirrors

Both a live backup and a transaction log mirror provide a secondary copy of the transaction log. However, there are several differences between using a live backup and using a transaction log mirror:

A transaction log mirror is enabled when the database is initialized

Unlike a live backup, which is started on a live database using the Backup utility (dbbackup), transaction log mirroring is enabled at the time the database is initialized, using the -m server option.

A live backup is typically made to a different computer

By running the Backup utility (dbbackup) on a separate computer, the database server does not do the writing of the backed up transaction log file, and the data transfer is done by the SQL Anywhere client/server communications system. Therefore, the performance impact is decreased and reliability is greater.

Running a transaction log mirror on a separate computer is not recommended. It can lead to performance and data corruption problems, and stops the database server if the connection between the computers fails.

A live backup provides protection against a computer becoming unusable

Even if a transaction log mirror is kept on a separate device, it does not provide immediate recovery if the whole computer becomes unusable. You could consider an arrangement where two computers share access to a set of disks.

A live backup can lag behind the database server

A transaction log mirror contains all the information required for complete recovery of committed transactions. Depending on the load that the database server is processing, the live backup may lag behind the transaction log mirror and may not contain all the committed transactions.

Related Information

Transaction Log Mirrors [page 301]
Restarting a Database from a Live Backup [page 978]

1.6.1.1.1.2 SQL Anywhere Volume Shadow Copy Service (VSS)

SQL Anywhere is compatible with the Microsoft Volume Shadow Copy Service (VSS).

By default, all SQL Anywhere databases can use the VSS service for backups if the SQL Anywhere VSS writer (`dbvss17.exe`) is running. You can use VSS without the SQL Anywhere VSS writer to back up databases. However, you might need to use the full SQL Anywhere recovery procedures to restore those databases.

To prevent a database server from participating in the VSS service, include -vss- when starting the database server. You can also control when the VSS service is running using the Service utility (dbsvc) for Windows.

How VSS works with SQL Anywhere:

1. Your backup application sends a command to VSS to take a snapshot.
2. VSS issues an *identify* command to the SQL Anywhere VSS writer (`dbvss17.exe`).
3. VSS issues a *prepare to snapshot* command to suspend all transactions and write all modified pages to disk on all databases on all database servers. If transactions are not suspended on a database within 10 seconds, the snapshot might contain uncommitted transactions and full recovery may be necessary.
4. VSS issues a *freeze* command to checkpoint and then suspend all activity on all databases on all database servers. Each SQL Anywhere database server waits a maximum of 60 seconds for all databases to suspend all activity. Typically, this process takes a few seconds.
5. VSS issues a *thaw* command to the SQL Anywhere VSS writer to resume all transactions on all databases on all database servers.

Even if you use the VSS service, consider using a backup strategy that involves full backups and periodic incremental backups. In rare circumstances, SQL Anywhere might be unable to suspend transactions or complete a checkpoint within the maximum time allowed by VSS. If this situation occurs, and you have been performing SQL Anywhere backups, use the transaction log file and the full recovery process to recover your data.

Related Information

Creating Windows Services (SQL Central) [page 356]
Service Utility (dbsvc) for Windows [page 1212]

1.6.1.1.1.3 Parallel Database Backups

Parallel backups use physical device-level parallelism to decrease the overall time required to complete a backup operation.

When you perform a server-side image backup by using the Backup utility (dbbackup) with the -s option, or by using the BACKUP DATABASE statement with the AUTO TUNE WRITERS clause, a parallel database backup is performed.

The database server creates a reader thread for each drive on which database files are stored. A writer thread is created for the destination drive where the backup directory is located. Using separate readers and writers allows I/O operations to be performed in parallel, instead of sequentially.

The performance of a parallel backup is limited by the slowest component in the system. It is typically a physical disk, but it could also be other components, such as the I/O controller or the system bus. Each of these components has a maximum rate at which it can transfer data.

The BACKUP DATABASE statement and the Backup utility (dbbackup) provide options that let you configure the behavior of a parallel backup, including:

- when and how the checkpoint log is copied
- the maximum number of pages used at a time to transfer data from the database server to dbbackup (only available when using dbbackup)
- adding more writers (BACKUP DATABASE statement only)

Backups should always be made to a separate physical drive. This practice provides a performance benefit from the I/O parallelism, and also improves the safety of the data in the event of a hardware failure.

Related Information

BACKUP DATABASE Statement
Backup Utility (dbbackup) [page 1121]

1.6.1.1.2 Backup Tasks

There are several tasks associated with creating and managing backups.

In this section:

Backing up a Database Using the Backup Utility (dbbackup) [page 946]
Backup a database using the Backup utility (dbbackup).

Backing up a Database Using the BACKUP DATABASE Statement [page 949]
Backup a database using the BACKUP DATABASE statement.

Backing up a Database Using SQL Central [page 951]
Use wizards in SQL Central to perform the various types of backup.

1.6.1.1.2.1 Backing up a Database Using the Backup Utility (dbbackup)

Backup a database using the Backup utility (dbbackup).

Prerequisites

You must have the BACKUP DATABASE and VALIDATE ANY OBJECT system privileges.

If disk sandboxing is enabled and you are backing up to a location outside of the sandbox, you need the secure feature key for the database server.

Context

The Backup utility (dbbackup) requires a connection to the database being backed up. Therefore, it is only used for online backups. However, validating a table or an entire database should be performed while no connections are making changes to the database; otherwise, errors may be reported indicating some form of database corruption even though no corruption actually exists.

Choose a backup location that is not on the same computer as the location of the database. This practice provides the best protection in the event of media failure.

You cannot perform an archive backup using the Backup utility (dbbackup).

Procedure

1. Validate the database. For example, execute a dbvalid command similar to the following:

   ```
   dbvalid -c "UID=DBA;PWD=sql;SERVER=demo17;DBN=demo"
   ```

2. Execute a dbbackup command similar to one of the following examples. The examples in the Syntax column are split over multiple lines. However, the command you execute must be one line.

Option	Action
• **full** • **image** • **client-side**	`dbbackup -c "HOST=myHost;SERVER=demo17;DBN=demo; UID=DBA;PWD=sql" "c:\temp\SQLAnybackup"`
• **full** • **image** • **server-side** • **wait for trans-actions to complete**	`dbbackup -c "HOST=myHost;SERVER=demo17;DBN=demo; UID=DBA;PWD=sql"-s -wa "c:\temp\SQLAnybackup"`
• **full** • **image** • **client-side**	`dbbackup -c "HOST=myHost;SERVER=demo17;DBN=demo;UID=DBA;PWD=sql" "c:\temp\SQLAnybackup"`
• **incremental** • **image** • **client-side**	`dbbackup -c "HOST=myHost;SERVER=demo17;DBN=demo; UID=DBA;PWD=sql"-t -r -n "c:\temp\SQLAnybackup"`
• **live**	**full backup**: `dbbackup -c "HOST=myHost;SERVER=demo17;DBN=demo" -wa "c:\temp\SQLAnybackup";` **live backup**: `dbbackup -c "HOST=myHost;SERVER=demo17;DBN=demo" "c:\temp\SQLAnybackup" -l "demo.log"`

Results

The backup is performed, and an entry is made in the `backup.syb` configuration file.

Example

This example creates a live backup of the sample database, demo. The first command creates the full backup; a required step before creating a live backup, since recovery requires a database file to apply the backup transaction log file to. The second commands initiates the live backup.

```
dbbackup -c "SERVER=demo17;DBN=demo;UID=DBA;PWD=sql" "c:\temp\SQLAnybackup"
dbbackup -c "SERVER=demo17;DBN=demo;UID=DBA;PWD=sql" "c:\temp\SQLAnybackup" -l
"demo.log"
```

When the live backup starts, a message is displayed indicating that the live backup of the transaction log is waiting for the next page. Depending on your operating system, it may look like there is no activity on the backup transaction log file (*demo*`.log`). This is because the file remains open for the duration of the live backup, and some operation systems do not modify the metadata for a file (size, last modified timestamp, and so on) until the file is closed. In this case, the file is closed when the live backup is terminated, for example after a failure of the active database, or if the connection to the database is lost.

In the -l option, if the transaction log file name is the same as the file created during the full backup, you are asked to confirm overwriting the file. If you specify a different name, a new backup transaction log is created with that name and used for the live backup. If your database fails, the live backup stops. You can then apply the backup transaction log to the backup database you created during the full backup, assuming the active transaction log has not been renamed.

> **i** Note
>
> Recovery from this live backup example will only succeed as long as the active transaction log is never renamed.

Next Steps

Test your ability to recover from the backup you created.

Related Information

Live Backups (Continuous Backups) [page 944]
Types of Backup [page 941]
Database Recovery [page 960]
Disk Sandboxing [page 1687]
Backup Utility (dbbackup) [page 1121]

Validation Utility (dbvalid) [page 1257]

1.6.1.1.2.2 Backing up a Database Using the BACKUP DATABASE Statement

Backup a database using the BACKUP DATABASE statement.

Prerequisites

You must have the BACKUP DATABASE and VALIDATE ANY OBJECT system privileges.

If disk sandboxing is enabled, and you are backing up to a location outside of the sandbox, you need the secure feature key for the database server.

Validating a table or an entire database should be performed while no connections are making changes to the database; otherwise, errors may be reported indicating some form of database corruption even though no corruption actually exists.

Context

The BACKUP DATABASE statement is run on the database server computer. Since it requires a connection to a running database, it performs only server-side, online backups.

When backing up to disk, choose a location that is not on the same computer as the location of the database. This provides the best protection in the event of media failure of the machine on which the database is running.

Procedure

1. Validate the database. For example, execute a VALIDATE statement similar to the following:

   ```
   VALIDATE DATABASE;
   ```

2. Execute a BACKUP DATABASE statement, similar to one of the following examples.

Option	Action
• **full** • **image**	`BACKUP DATABASE DIRECTORY 'c:\\temp\\SQLAnybackup';`
• **incremental** • **image**	`BACKUP DATABASE DIRECTORY 'c:\\temp\\SQLAnybackup'` `TRANSACTION LOG ONLY TRANSACTION LOG TRUNCATE;`
• **full**	`BACKUP DATABASE TO 'c:\\temp\\SQLAnybackup\\archive';`

Option	Action
• archive • to disk	
• full • image • transaction log renamed	`BACKUP DATABASE DIRECTORY 'c:\\temp\\SQLAnybackup' TRANSACTION LOG RENAME;`
• incremental • image • transaction log renamed	`BACKUP DATABASE DIRECTORY 'c:\\temp\\SQLAnybackup' TRANSACTION LOG ONLY TRANSACTION LOG RENAME MATCH;`

Results

The backup is performed, and an entry is made in the `backup.syb` configuration file.

Example

The following statement creates an event that backs up the database once a day, using a dynamically constructed directory name to place the backup in.

```
CREATE EVENT NightlyBackup
SCHEDULE
START TIME '23:00' EVERY 24 HOURS
HANDLER
BEGIN
 DECLARE dest LONG VARCHAR;
 DECLARE day_name CHAR(20);
 SET day_name = DATENAME( WEEKDAY, CURRENT DATE );
 SET dest = 'd:\\temp\\SQLAnybackup\\' || day_name;
 BACKUP DATABASE DIRECTORY dest
 TRANSACTION LOG RENAME;
END;
```

Next Steps

Test your ability to recover from the backup you created.

Related Information

Types of Backup [page 941]
Database Recovery [page 960]
Disk Sandboxing [page 1687]

BACKUP DATABASE Statement
VALIDATE Statement

1.6.1.1.2.3 Backing up a Database Using SQL Central

Use wizards in SQL Central to perform the various types of backup.

Prerequisites

You must have the BACKUP DATABASE and VALIDATE ANY OBJECT system privileges.

Validating a table or an entire database should be performed while no connections are making changes to the database; otherwise, errors may be reported indicating some form of database corruption even though no corruption actually exists.

Context

Use the *Create Image Backup* to create an image backup that creates copies of each file associated with a database.

Use the *Backup Database Wizard* to an archive backup (a single file) containing the database and any associated files.

If disk sandboxing is enabled and you are backing up to a location outside of the sandbox, then you need the secure feature key for the database server.

When backing up to disk, choose a location that is not on the same computer as the location of the database. This practice provides the best protection in the event of media failure of the computer on which the database is running.

Procedure

1. Validate the database:
   a. Right-click the database and click *Validate Database*.
   b. Follow the instructions in the *Validate Database Wizard*. However, on the *Choose Validation Types* screen, ensure that the following options are selected:
      - Select *Validate database pages*, and choose *Full check*.
      - Select *Validate tables and materialized views*, and choose *Normal check*.
2. Back up the database:

Option	Action
Image backup	1. Right-click the database, click *Create Image Backup*, and follow the instructions in the wizard. 2. When asked which files you want to backup: • For a full backup, click *All database files and the transaction log file*. • For an incremental backup, click *Transaction log file only*.
Archive backup	1. Right-click the database, click *Backup Database*, and follow the instructions in the wizard. You can only perform a full backup when creating an archive backup.

Results

The backup is performed, and an entry is made in the `backup.syb` configuration file.

Next Steps

Test your ability to recover from the backup you created. Otherwise, if your database fails and you are unable to recover from the backup, you may lose data.

Related Information

Types of Backup [page 941]
Database Recovery [page 960]
Restoring a Database from an Archive Backup (SQL Central) [page 976]

1.6.1.1.3 Design a Backup and Recovery Strategy

You can implement a backup and recovery strategy to protect your data.

A good backup strategy incorporates a combination of full and incremental backups, and possibly a live backup as well. Also, validate your database before backing up, and validate your backups in read-only mode to ensure that they can be used for recovery. The frequency with which you make backups depends on such factors as the importance of your data and how often it changes. A common starting point is to perform a weekly full backup, with daily incremental backups of the transaction log. Both full and incremental backups can be performed while the database is running (online).

Some of the factors that you need to consider when developing your backup and recovery plan include:

- what files need to be backed up
- where the database files are located

- where the backup files are going to be stored
- how many full backups you plan to keep around. If you overwrite a backup with a new backup, then a media failure in the middle of the backup leaves you with no backup at all. Keep multiple copies of full backups around.
- whether the database server needs to remain available while the backup is running. Keep some of your full backups off site to protect against fire, flood, earthquake, theft, or vandalism.
- how long your organization can function without access to the database (maximum recovery time in the event of media failure). External factors such as available hardware, the size of database files, recovery medium, disk space, and unexpected errors can affect your recovery time. When planning a backup strategy, allow additional recovery time for tasks such as entering recovery commands or retrieving and loading tapes, if needed.
- whether the database involved in replication (if so, consider using a transaction log mirror).

High-Level Steps for Creating a Backup and Recovery Strategy

1. Create and verify your backup and recovery commands, including commands for database validation.
2. Measure the time it takes to execute the backup and recovery commands.
3. Write down the backup commands and create written procedures outlining where your backups are kept. The procedures should identify any naming conventions that are used and the type of backups that are performed.
4. Automate/schedule your backup procedures on the production server using **maintenance plans**.
5. Monitor backup procedures to avoid unexpected errors. Document any changes in the process.

In this section:

Related Information

Types of Backup [page 941]
Task Automation Using Schedules and Events [page 997]
Considerations for Databases Involved in Synchronization and Replication [page 959]

1.6.1.1.3.1 Maintenance Plans Automate Validation and Backup

A maintenance plan is a schedule to run one or more maintenance tasks automatically.

You configure maintenance plans in SQL Central, and the types of maintenance tasks you can configure are:

- validation
- backup
- user-defined operations (SQL statements) that precede a validation task or follow a backup task

Although you can create many maintenance plans, only one maintenance plan can run at a time. Each time that a maintenance plan runs, a maintenance plan report is stored in the database. View the maintenance plan report in SQL Central or have the report emailed to you.

In this section:

1.6.1.1.3.1.1 Creating a Maintenance Plan

Create a maintenance plan using SQL Central.

Prerequisites

To create a maintenance plan, you must have the SELECT ANY TABLE system privilege.

You must also have the MANAGE ANY EVENT system privilege, and either the INSERT ANY TABLE system privilege or the INSERT privilege on the maint_plan table.

If you are running this task for the first time, or if the database has been upgraded since the last time the maintenance plans were accessed, you receive an error indicating that the maintenance plan tables must be created or updated. To create or update these tables, you must have the SELECT ANY TABLE system privilege, and either the CREATE ANY OBJECT system privilege or both the CREATE ANY INDEX and CREATE ANY TABLE system privileges. Click the *Maintenance Plans* folder to create or update the tables.

Procedure

1. In the left pane of SQL Central, right-click *Maintenance Plans* and click *New* > *Maintenance Plan*.
2. Follow the instructions in the wizard.

Results

A maintenance plan is created and runs using the specified schedule.

Related Information

Types of Backup [page 941]

1.6.1.1.3.1.2 Viewing a Maintenance Plan Report

View a maintenance plan report in SQL Central.

Prerequisites

To view a maintenance plan report, you must have the SELECT ANY TABLE system privilege.

If the database has been upgraded since the last time the maintenance plans were accessed, you receive an error indicating that the maintenance plan tables must be updated. To update these tables, you must have the SELECT ANY TABLE system privilege, and either the CREATE ANY OBJECT system privilege or both the CREATE ANY INDEX and CREATE ANY TABLE system privileges.

Procedure

1. In the left pane of SQL Central, click *Maintenance Plans*.
2. In the left pane, click the maintenance plan.
3. In the right pane, double-click the report.

Results

The *Maintenance Plan Properties* window appears. The *Details* pane contains the report for the maintenance plan.

Related Information

Types of Backup [page 941]

1.6.1.1.3.1.3 Deleting a Maintenance Plan

Delete a maintenance plan using SQL Central.

Prerequisites

To delete a maintenance plan, you must have the SELECT ANY TABLE system privilege.

You must also have the MANAGE ANY EVENT system privilege and the DELETE ANY TABLE system privilege.

If the database has been upgraded since the last time the maintenance plans were accessed, then you receive an error indicating that the maintenance plan tables must be updated. To update these tables, you must have the SELECT ANY TABLE system privilege, and either the CREATE ANY OBJECT system privilege or both the CREATE ANY INDEX and CREATE ANY TABLE system privileges.

Procedure

1. In the left pane of SQL Central, click *Maintenance Plans*.
2. Right-click the maintenance plan and click *Delete*.
3. Click *Yes* to confirm the deletion.

Results

The maintenance plan is deleted, and any validations or backups that ran as part of the maintenance plan do not run again.

Related Information

Types of Backup [page 941]

1.6.1.1.3.2 Validations in Your Backup and Recovery Plan

Validate your database periodically to ensure that backups you are creating are not corrupted.

A best practice is to create a maintenance plan that looks after regular database validation automatically.

If validation finds that a base table in the database file is corrupt, then treat the event as a media failure and recover from your previous backup. However, if the corruption is in an index, then it would be easier to drop and recreate the index.

Validation requires exclusive access to the object being validated, so it is best to validate when there is no other activity on the database.

There are a few situations where performing validation helps ensure you can recover from a backup:

Between backups

Database file corruption may not be confirmed until the database server tries to access the affected part of the database. As part of your backup and recovery plan, periodically validate your database between backups.

Before backup

All backup procedures should include a validation step that takes place prior to the backup.

After backup

You should validate your backup just after creating it. When you start a backup copy of a database to validate it, use the -ds database option to specify the location of dbspace files and the transaction log. This allows you to start the backed up copy of the database on the same computer as the original database while the original database is still running.

Validating a Database That Had Open Transactions at the Time of Backup

Start the database backup using the in-memory validation mode and then validate the database. In-memory mode allows the database server to perform recovery on the database while preventing any writes to the database.. You cannot use the read-only mode (with the -r database option) when there are pending transactions in the database.

Related Information

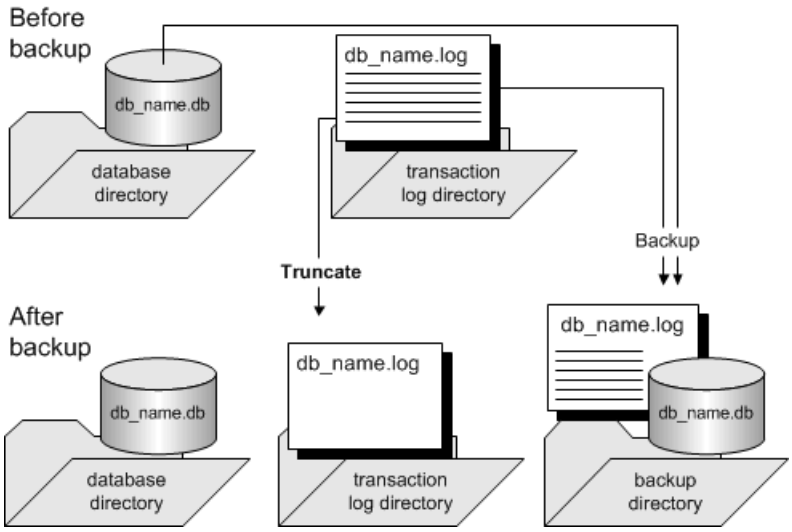
Database Validation [page 988]
Maintenance Plans Automate Validation and Backup [page 954]
Transaction Log Validation [page 294]

1.6.1.1.4 What Happens to the Transaction Log During Backup

When you make a backup, unless you specify otherwise, the database server makes a copy of the transaction log and leaves the transaction log in place to continue growing.

Disk space limitations can make it impractical to let the transaction log grow indefinitely. Choosing to delete the contents of the transaction log when the backup is complete, frees the disk space. However, do not choose this option if the database is involved in replication because replication requires access to the transaction log.

A full backup, which truncates the transaction log file, is illustrated in the figure below.



A backup copy of the transaction log is almost always smaller than the online transaction log. This difference happens because database server allocates space to the active transaction log in multiples of 64 KB, so the file size includes empty pages. However, only non-empty pages are backed up, accounting for the reduced file size of the backup copy.

Deleting the transaction log after each incremental backup makes recovery from a media failure on the database file a more complex task. Each transaction log must be applied in sequence to bring the database up to date, and there may then be several different transaction logs since the last full backup.

Related Information

Types of Backup [page 941]
Backing up a Database Using the Backup Utility (dbbackup) [page 946]
Backing up a Database Using the BACKUP DATABASE Statement [page 949]
Backing up a Database Using SQL Central [page 951]

1.6.1.1.5 Considerations for Databases Involved in Synchronization and Replication

For synchronization and replication environments, use backup options to rename and restart the transaction log.

This practice prevents open-ended growth of the transaction log, while maintaining information about old transactions.

If your database is part of a SQL Remote installation, then the Message Agent must have access to old transactions. If it is a consolidated database, it holds the master copy of the entire SQL Remote installation, and thorough backup procedures are essential to ensure that no data is lost.

If your database is participating in a MobiLink setup using dbmlsync, then the same considerations apply. However, if your database is a MobiLink consolidated database, then old transaction logs are not required.

Backup procedures are not as crucial on remote databases as they are on the consolidated database. You may choose to rely on replication to the consolidated database as a data backup method. In the event of a media failure, the remote database would have to be re-extracted from the consolidated database, and any operations that have not been replicated would be lost. You could use the Log Translation utility to attempt to recover lost operations.

Even if you do choose to rely on replication to protect remote database data, backups may still need to be done periodically at remote databases to prevent the transaction log from growing too large. You should use the same option (rename and restart the transaction log) as at the consolidated database, running the Message Agent so that it has access to the renamed log files. If you set the delete_old_logs option to On at the remote database, then the old transaction log files are deleted automatically by the Message Agent when they are no longer needed.

Automatic Transaction Log Renaming in SQL Remote

Use the -x Message Agent option to eliminate the need to rename the transaction log on the remote computer when the database server shuts down. The -x option renames the transaction log after it has been scanned for outgoing messages.

Related Information

Types of Backup [page 941]
Renaming or Truncating the Transaction Log (SQL Central) [page 296]
Log Translation Utility (dbtran) [page 1187]
delete_old_logs Option [MobiLink][SQL Remote] [page 729]
SQL Remote Message Agent Utility (dbremote)

1.6.1.2 Database Recovery

Recovery is the process of restoring your database file, transaction log, and dbspaces, and bringing the database file as up-to-date as possible with incremental transaction log files.

Always validate your backup as part of your backup and recovery plan; only recover from a valid backup copy of the database.

Automatic Recovery

On database startup, the database server checks whether the database shut down cleanly at the end of the previous session. If it did not, then the database server runs an automatic recovery process to restore all changes up to the most recently committed transaction. Backup copies of the database and transaction log must not be changed in any way prior to recovery.

Manual Recovery

Recover a database to a previous version of a database by applying one or more transaction logs to a copy of a database backup. The steps you take in the recovery process depend on whether you leave the transaction log untouched on incremental backup in your backup process. If your backup operation deletes or renames the transaction log, then you may have to apply changes from several transaction logs. If your backup operation leaves the transaction log untouched, then you use only the online transaction log in recovery.

If you have multiple transaction logs, then transactions may span several transaction logs. Apply the transaction logs in the correct order when recovering; otherwise, transactions that span multiple transaction logs are rolled back. Specify the -ad database server option if you want the database server to determine the correct order in which to apply the transaction logs.

You can also restore your database by performing a **point-in-time recovery (PITR)**. Operations performed by the database server are uniquely identified in the transaction log by an associated offset number. COMMIT and CHECKPOINT operations in the transaction log also have timestamps associated with them. You can perform a point-in-time recovery to an offset or to a timestamp in the transaction log. Restoring to an offset is ideal when you want more precision; you can specify the transaction you want to restore to.

Performance During Database Recovery

Database recovery can take a long time. Here are some of the factors that affect the time it takes to recover a database:

The concurrency level of the workload The greater the level of concurrency of the workload the better the recovery time.

The hardware resources that are available on the machine performing the recovery The more CPU cores and disk I/O spindles available, the less time required for recovery.

The mix of DDL and DML operations in the transaction log DML operations are better candidates to benefit from parallel recovery than DDL operations, which must be serialized, adding to the time it takes to recover the database.

Existence of primary keys Operations on tables with primary keys greatly speed up the recovery process.

In this section:

Related Information

Point-in-time Recovery [page 962]

1.6.1.2.1 The Automatic Recovery Process

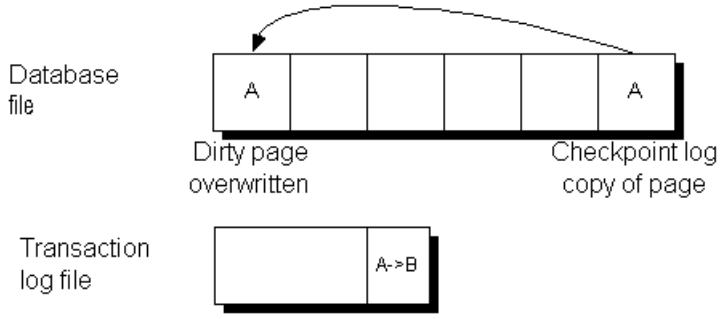
When a database is shut down, the database server performs a checkpoint so that information in the database is held in the database file.

This process is a called a clean shutdown.

Each time that you start a database, the database server checks whether the last shutdown was clean or the result of a system failure. If the database did not shut down cleanly, then it automatically takes the following steps to recover from a system failure:

Recover to the most recent checkpoint

To restore all pages to their state at the most recent checkpoint, the checkpoint log pages are copied over the changes made since the checkpoint.



Apply changes made since the checkpoint

Changes made between the checkpoint and the system failure, which are held in the transaction log, are applied.

Roll back uncommitted transactions

Any uncommitted transactions are rolled back, using the rollback logs.

1.6.1.2.2 Point-in-time Recovery

Point-in-time recovery (PITR) offers you a way to restore a database to a moment in time specified either as a timestamp value or a transaction log offset.

Operations are recorded sequentially in the transaction log, thus every operation can be uniquely identified by a numeric offset within the file. If you can identify an operation in the log up to which, and including which, the database was fine, then recovering to the nearest COMMIT or CHECKPOINT operation after that offset is recommended. This is known as **point-in-time recovery to an offset**.

COMMIT and CHECKPOINT operations in the transaction log have a timestamp recorded for when they were performed. If you know a time before which your database was working fine, you can restore to that time. The database server then recovers to the nearest COMMIT or CHECKPOINT after the specified timestamp. This is known as **point-in-time recovery to a timestamp**.

The recommended approach when performing point-in-time recovery is to use the Translation Log utility (dbtran) to examine operations in the transaction log and then recover to the operation using the offset.

Point-in-time recovery is initiated in the same way as a typical recovery operation, by specifying one of the -a options (-a, -ad, -ar, and so on). The database server can find and apply transactions from applicable transaction log files in more than one directory (-ad database server option).

Point-in-time recovery differs from other recovery methods in one significant way. You can only apply transaction logs all at once during the recovery. If at the end of the recovery you discover another transaction log with transactions that should be applied, then you must perform the recovery again and include the location of the new transaction log in the list of directories specified in the -ad clause.

At the end of a point-in-time recovery, the database displays the timestamp and offset of the last COMMIT operation that was performed.

Related Information

Database Recovery [page 960]
Tutorial: Restoring the Database to an Offset in the Transaction Log [page 970]
Tutorial: Restoring the Database to a Timestamp [page 965]
-ad Database Option [page 545]
-ru Database Option [page 557]
-ruo Database Option [page 558]

1.6.1.2.3 Recovering Uncommitted Operations (SQL Central)

Use SQL Central to recover uncommitted operations and reapply all committed transactions to the database.

Context

In some circumstances, you need to find information about transactions that were incomplete at the time of the failure. When recovering from media failure on the database file, the transaction log must be intact.

The *Translate Log File Wizard* helps you translate a transaction log file into a `.sql` file from SQL Central.

> **i** Note
>
> The transaction log may or may not contain changes right up to the point where a failure occurred. It does contain any changes made before the end of the most recently committed transaction that made changes to the database.

Procedure

1. In SQL Central, click *Tools* > *SQL Anywhere 17* > *Translate Log File*.
2. Follow the instructions in the wizard.
3. Edit the translated transaction log (SQL script file) in a text editor and identify the instructions that you need.

Results

The uncommitted operations are recovered.

Related Information

Database Recovery with Multiple Transaction Logs [page 979]
Recover from Media Failure [page 983]
Restoring from an Image Backup [page 975]
Restoring a Database from an Archive Backup (SQL Central) [page 976]
Restarting a Database from a Live Backup [page 978]

1.6.1.2.4 Recovering Uncommitted Operations (Command Line)

Use the dbtran utility to recovery uncommitted operations and reapply all committed transactions to the database.

Context

In some circumstances, you need to find information about transactions that were incomplete at the time of failure. When recovering from media failure on the database file, the transaction log must be intact.

> **i** Note
>
> The transaction log may or may not contain changes up to the point where a failure occurred. It contains any changes made before the end of the most recently committed transaction that made changes to the database.

Procedure

1. Run dbtran to convert the transaction log file into a SQL script file, using the -a option to include uncommitted transactions.
2. Edit the translated transaction log (SQL script file) in a text editor and identify the instructions you need.

Results

The uncommitted operations are recovered.

Example

Run the following command to translate the transaction log file `sample.log` into `changes.sql` file:

```
dbtran -a sample.log changes.sql
```

Related Information

Database Recovery with Multiple Transaction Logs [page 979]
Recover from Media Failure [page 983]
Restoring from an Image Backup [page 975]
Restoring a Database from an Archive Backup (SQL Central) [page 976]
Restarting a Database from a Live Backup [page 978]
Log Translation Utility (dbtran) [page 1187]

1.6.1.2.5 Tutorial: Restoring the Database to a Timestamp

Restore a database to a timestamp in the transaction log.

Prerequisites

You must have the CREATE TABLE system privilege to create the table used in this tutorial.

You must have the BACKUP DATABASE system privilege to perform the backup steps in this tutorial.

The privileges required to perform a database restore depend on the setting for the -gu database option, and whether you have the SERVER OPERATOR system privilege.

Context

You can restore a database to an approximate point in time. For example, if you know your database became unusable after a certain point in time, then you can recover to a timestamp just prior to that time.

When you restore to a timestamp, the database server restores to the approximate COMMIT operation associated with that timestamp. Point-in-time recovery to a timestamp gives you less precision than point-in-time recovery to an offset in the transaction log.

When executing a restore command, if the restore is not successful, then the copy of the database you are trying to restore becomes unusable for the restore. It is a best practice to work with *copies* of your backup files, instead working with the backup files themselves; you can make new copies.

To save time and effort, add the `-im v` option to your restore command (for example, `dbsrv17 -im v myDB.db -as -ar -ru "2014-12-15 15:23:15.348"`). This option allows you to test if the restore command runs successfully, but does not permanently change the database. If the restore is not successful, then you do not have to create new copies of the backup files; you can alter the restore command and run it again. If the command is successful, then you can connect to the read-only restored database and confirm that the database was restored to the right point in time. If the database was restored to the correct point in time, then shut down the database and run the restore command again without the `-im v` option to permanently restore the database to the desired point in time.

Procedure

1. In *Interactive SQL*, connect to the sample database (demo.db).
2. Execute the following statement to create a backup directory, `c:\temp\backup_dir\`, and back up the database file to it:

   ```
   BACKUP DATABASE DIRECTORY 'c:\\temp\\backup_dir' DBFILE ONLY;
   ```

   The `c:\temp\backup_dir\` directory now contains a backed up copy of the demo.db database.
3. Execute the following statement to create a table called trackingTable that you will use later to confirm which timestamp you restored the database to. The INSERT statement sets the initial value, and the COMMIT statement commits the changes to the database.

   ```
   CREATE TABLE trackingTable ( myValue INT );
   INSERT INTO trackingTable VALUES (1);
   COMMIT;
   ```
4. Execute the following statement to return the timestamp of the current time. In this example, the timestamp is 2014-12-15 15:23:15.348. Your timestamp will be different; record it for later use.
5. To format the timestamp value, click *Tools* > *Options*, and then select the *SQL Anywhere* category. Set *Format dates and times setting* to *Locally*, and *How do you want results to be displayed?* to *Text*.

   ```
   SELECT CURRENT TIMESTAMP;
   ```

   Example result:

   ```
   current timestamp
   ----------------------
   ```

```
2014-12-15 15:23:15.348
(1 rows)
```

> **i Note**
>
> To simplify this tutorial, you are determining the timestamp to restore your database to by querying the CURRENT TIMESTAMP special value and noting the time you want to restore the database to. However, in a real database restore situation, you would either use the Log Translation utility (dbtran) to help you determine the timestamp you want to restore the database to, or specifying a timestamp that coincides generally with the time you want to restore the database.

6. Execute the following statement to back up the transaction log to the directory you backed up the database to in an earlier step:

   ```
   BACKUP DATABASE DIRECTORY 'c:\\temp\\backup_dir' TRANSACTION LOG ONLY
   TRANSACTION LOG RENAME MATCH;
   ```

   This statement creates a backup of the transaction log file that has a name that is something like `<integer>AA.log` (for example, `141212AA.LOG`).

7. Execute the following statements to add a row in the trackingTable table and commit the changes to the database:

   ```
   INSERT INTO trackingTable VALUES ( 2 );
   COMMIT;
   ```

8. Execute the following statement to return the timestamp of the current time. In this example, the timestamp is 2014-12-15 15:24:14.922. Yours will be different, though; note it down.

   ```
   SELECT CURRENT TIMESTAMP;
   ```

   Example result:

   ```
   current timestamp
   ----------------------
   2014-12-15 15:24:14.922
   (1 rows)
   ```

9. Execute the following statement to confirm there are two rows in the trackingTable table:

   ```
   SELECT * FROM trackingTable;
   ```

   Example result:

   ```
   myValue
   -----------
             1
             2

   (2 rows)
   ```

10. Execute the following statement to create another backup of the transaction log in the directory where you backed up the database:

    ```
    BACKUP DATABASE DIRECTORY 'c:\\temp\\backup_dir' TRANSACTION LOG ONLY
    TRANSACTION LOG RENAME MATCH;
    ```

This creates a backup of the log file that has a name that is something like `<integer>AB.log` (for example, `141212AB.LOG`).

In your `c:\temp\backup_dir` directory, you should now have one backup copy of the database (demo.db), and two uniquely named backup copies of the log file. You are now ready to restore your database to the first timestamp you noted down earlier.

11. Disconnect from, and shut down, your database.
12. Copy your `c:\temp\backup_dir` to a new directory called `c:\temp\restore_dir`. This is the directory you will work from to restore the database to a timestamp.

    > **i Note**
    >
    > It is a best practice to restore from a copy of your backup files, instead of directly from the backup files. That way, if a problem occurs, you can make new copies and start the restoration attempt again.

13. In `c:\temp\restore_dir`, execute the following command to restore the database to the first timestamp you noted during this tutorial (in this example, 2014-12-15 15:23:15):

    ```
    dbsrv17 demo.db -as -ar -ru "2014-12-15 15:23:15 -05:00"
    ```

    Including the time zone (-05:00) in the timestamp specification is a best practice and highly recommended when doing point-in-time recovery to a timestamp. In this case -05:00 tells the database server that the times in the transaction log are in UTC minus 5 hours. When you omit the time zone in the timestamp specification, the local time of the database server doing the restore is used. If the database server and the logs are in the same time zone, then this is not a problem. However, if they are not in the same time zone, then the database is restored to the timestamp interpreted in the local time of the database server, which is incorrect.
14. Connect to the demo.db database in `c:\temp\restore_dir\` using *Interactive SQL*, and execute the following statement to query the rows in the trackingTable table:

    ```
    SELECT * FROM trackingTable;
    ```

    Example result:

    ```
    myValue
    -----------
              1

    (1 row)
    ```

    The fact that there is only one row in the trackingTable table confirms that the restoration to the specified timestamp was successful because the first timestamp you noted occurred prior to inserting the second row (2) into the table.
15. Execute the following statements to add a row in the trackingTable table and commit the change to the database:

    ```
    INSERT INTO trackingTable VALUES ( 103 );
    COMMIT;
    ```

16. Execute the following statement to return the timestamp of the current time. In this example, the timestamp is 2014-12-15 15:54:23.84. Yours will be different; record it for later use.

    ```
    SELECT CURRENT TIMESTAMP;
    ```

Example result:

```
current timestamp
-----------------------
2014-12-15 15:54:23.84
(1 rows)
```

17. Execute the following statements to add a row in the trackingTable table and commit the changes to the database:

```
INSERT INTO trackingTable VALUES ( 104 );
COMMIT;
```

18. Execute the following statement to return the timestamp of the current time. In this example, the timestamp is 2014-12-15 15:55:33.048. Yours will be different; record it for later use.

```
SELECT CURRENT TIMESTAMP;
```

Example result:

```
current timestamp
-----------------------
2014-12-15 15:55:33.048
(1 rows)
```

19. Execute the following statement to back up the transaction log to the directory you backed up the database to in an earlier step.

```
BACKUP DATABASE DIRECTORY 'c:\\temp\\backup_dir' TRANSACTION LOG ONLY
TRANSACTION LOG RENAME MATCH;
```

This statement creates a third backup of the log file in `c:\temp\backup_dir`.

20. Disconnect from the database and stop the database server.
21. Copy the contents of `c:\temp\backup_dir` to a new directory called `c:\temp\restore_dir2` (note the 2 at the end).
22. In `c:\temp\restore_dir2`, run the following command to restore the database to the timestamp you noted after adding the value 103 to trackingTable.myValue (in this example, 2014-12-15 15:54:23.84):

```
dbsrv17 demo.db -as -ar -ru "2014-12-15 15:54:23.84 -05:00"
```

23. In *Interactive SQL*, connect to the demo.db database in `c:\temp\restore_dir2`.
24. Execute the following statement to view the rows in the trackingTable table:

```
SELECT * FROM trackingTable;
```

Example result:

```
myValue
-----------
          1
          103
(2 rows)
```

Notice that the value 2 from the older log, and the value 104 from the new log, are not included. This is because the value 2 was inserted after the first backup operation and therefore wasn't present in the

restored database in `c:\temp\restore_dir\`. Likewise, the value 104 was inserted after the second backup operation and therefore wasn't present in the restored database in `c:\temp\restore_dir2\`.

25. Once your point-in-time recovery is complete, shut down the database and restart the database server without any of the backup-related options (for example, -a, -ad, and -ar).

Related Information

Point-in-time Recovery [page 962]
Database Recovery [page 960]
Tutorial: Restoring the Database to an Offset in the Transaction Log [page 970]
-ad Database Option [page 545]
-ru Database Option [page 557]
-ruo Database Option [page 558]

1.6.1.2.6 Tutorial: Restoring the Database to an Offset in the Transaction Log

Restore a database to an offset in the transaction log.

Prerequisites

You must have the CREATE TABLE system privilege to create the table used in this tutorial.

You must have the BACKUP DATABASE system privilege to perform the backup steps in this tutorial.

The privileges required to perform a database restore depend on the setting for the -gu database option, and whether you have the SERVER OPERATOR system privilege.

Context

Operations are recorded sequentially in the transaction log, thus every operation can be uniquely identified by a numeric offset within the transaction log. If you can identify an operation or event in the log up to which, and including which, the database was fine, then you can restore the database up to and including the offset of this operation.

When executing a restore command, if the restore is not successful, then the copy of the database you are trying to restore becomes unusable for the restore. For this reason it is a best practice to work with *copies* of your backup files, instead working with the backup files themselves; you can always make new copies.

To save time and effort, add the `-im v` option to your restore command (for example, `dbsrv17 -im v myDB.db -as -ar -ruo 1245672`). This option allows you to test if the restore command runs successfully,

but does not permanently change the database. If the restore is not successful, then you do not have to create new copies of the backup files; you can alter the restore command and run it again. If the command is successful, then you can connect to the read-only restored database and confirm that the database was restored to the right point in time. If the database was restored to the correct point in time, then shut down the database and run the restore command again without the `-im v` option to permanently restore the database to the desired point in time.

Procedure

1. In *Interactive SQL*, connect to the sample database (demo.db).
2. Execute the following statement to create a backup directory, `c:\temp\backup_dir\`, and back up the database file to it:

   ```
   BACKUP DATABASE DIRECTORY 'c:\\temp\\backup_dir' DBFILE ONLY;
   ```

   `c:\temp\backup_dir\` now contains a backed up copy of the sample database (demo.db).
3. Execute the following statement to create a table called trackingTable that you will use later to confirm which offset you restored the database to. The INSERT statement sets the initial value, and the COMMIT statement commits the changes to the database.

   ```
   CREATE TABLE trackingTable ( myValue INT );
   INSERT INTO trackingTable VALUES (1);
   COMMIT;
   ```
4. Execute the following statement to get the offset of the operation right after the COMMIT statement you executed from the transaction log. The resulting offset value in this example is 1245672. However, because your offset number is going to be different than the offset number returned in this tutorial, note your offset number; you will specify it later (instead of the offset number shown in this tutorial) when specifying the offset to restore the database to.

   ```
   SELECT DB_PROPERTY('LastCommitRedoPos');
   ```

   Example result:

   ```
   db_property('LastCommitRedoPos')
   ----------------------------------------
   1245672
   (1 rows)
   ```

   > **i Note**
   >
   > To simplify this tutorial, you are determining the offset to restore your database to by querying the LastCommitRedoPos database property to find out the offset of the last COMMIT statement. However, in a real database restore situation, you would use the Log Translation utility (dbtran) to help you determine the offset of the actual operation you want to restore the database to.
5. Execute the following statement to back up the transaction log to the directory you backed up the database to in an earlier step:

   ```
   BACKUP DATABASE DIRECTORY 'c:\\temp\\backup_dir' TRANSACTION LOG ONLY
   TRANSACTION LOG RENAME MATCH;
   ```

This statement creates a backup of the transaction log file that has a name that is something like `<date>AA.log` (for example, `141212AA.LOG`).

6. Execute the following statements to add a row in the trackingTable table and commit the change to the database:

```
INSERT INTO trackingTable VALUES ( 2 );
COMMIT;
```

7. Execute the following statement to get the offset of the COMMIT statement you executed from the transaction log. The resulting offset value in this example is 1249951. Again, your offset number is going to be different than the offset number returned in this tutorial; note your offset number.

```
SELECT DB_PROPERTY('LastCommitRedoPos');
```

Example result:

```
db_property('LastCommitRedoPos')
----------------------------------------
1249951
(1 rows)
```

8. Execute the following statement to confirm that there are two rows in the trackingTable table:

```
SELECT * FROM trackingTable;
```

Example result:

```
myValue
-----------
          1
          2

(2 rows)
```

9. Execute the following statement to create a second backup of the transaction log in the directory where you backed up the database:

```
BACKUP DATABASE DIRECTORY 'c:\\temp\\backup_dir' TRANSACTION LOG ONLY
TRANSACTION LOG RENAME MATCH;
```

This statement creates a backup of the transaction log file that has a name that is something like `<integer>AB.log` (for example, `141212AB.LOG`).

In `c:\temp\backup_dir\`, you should now have one backup copy of the database (demo.db), and two uniquely named backup copies of the log file. You are now ready to restore your database to the offsets you noted earlier.

10. Disconnect from the database and shut it down.
11. Copy `c:\temp\backup_dir\` to a new directory called `c:\temp\restore_dir\`. This is the directory you will work from to restore the database to an offset.

> **i Note**
>
> It is a best practice to restore from a copy of your backup files, instead of directly from the backup files. That way, if a problem occurs, you can make new copies and start the restoration attempt again.

12. In `c:\temp\restore_dir\`, run the following command to restore the database to the first offset you noted during this tutorial (in this example, offset 1245672):

    ```
    dbsrv17 demo.db -as -ar -ruo 1245672
    ```

    Notes:

    - The -as option tells the database server to leave the database running after restoring it.
    - The -ar option tells the database server to apply the logs from the local directory.
    - The -ruo option tells the database server to restore the database starting at offset 1245672.

13. In *Interactive SQL*, connect to the demo.db database in `c:\temp\restore_dir\` and execute the following statement to view the rows in the trackingTable table:

    ```
    SELECT * FROM trackingTable;
    ```

    Example result:

    ```
    myValue
    -----------
            1
    (1 row)
    ```

    The fact that there is only one row in the trackingTable table confirms that the restoration to the specified offset was successful because the first offset occurred prior to committing a second value (2) into the table.

14. Execute the following statements to add a row in the trackingTable table and commit the change to the database:

    ```
    INSERT INTO trackingTable VALUES ( 103 );
    COMMIT;
    ```

15. Execute the following statement to get the offset of the COMMIT statement you executed from the transaction log. The resulting offset value in this example is 1249890. Note your offset number.

    ```
    SELECT DB_PROPERTY('LastCommitRedoPos');
    ```

    Example result:

    ```
    db_property('LastCommitRedoPos')
    ----------------------------------------
    1249890
    (1 rows)
    ```

16. Execute the following statements to add a row in the trackingTable table and commit the change to the database:

    ```
    INSERT INTO trackingTable VALUES ( 104 );
    COMMIT;
    ```

17. Execute the following statement to get the offset of the COMMIT statement you executed from the transaction log. The resulting offset value in this example is 1249914. Note your offset number.

    ```
    SELECT DB_PROPERTY('LastCommitRedoPos');
    ```

    Example result:

```
db_property('LastCommitRedoPos')
-------------------------------------
1249914
(1 rows)
```

18. Execute the following statement to back up the transaction log to the directory you backed up the database to in an earlier step.

    ```
    BACKUP DATABASE DIRECTORY 'c:\\temp\\backup_dir' TRANSACTION LOG ONLY
    TRANSACTION LOG RENAME MATCH;
    ```

    This statement creates a third backup of the transaction log file in `c:\temp\backup_dir\`.
19. Disconnect from the database and stop the database server.
20. Copy your `c:\temp\backup_dir\` to a new directory called `c:\temp\restore_dir2` (note the 2 at the end).
21. In `c:\temp\restore_dir2\`, run the following command to restore the database to the offset you noted after adding the value 103 to trackingTable.myValue (in this example, 1249890):

    ```
    dbsrv17 demo.db -as -ar -ruo 1249890
    ```
22. In *Interactive SQL*, connect to the demo.db database in `c:\temp\restore_dir2\`.
23. Execute the following statement to view the rows in the trackingTable table:

    ```
    SELECT * FROM trackingTable;
    ```

    Example result:

    ```
    myValue
    -----------
              1
              103
    (2 rows)
    ```

    The value 2 from the older log, and the value 104 from the new log, are not included. This is because the value 2 was inserted after the first backup operation and therefore wasn't present in the restored database in `c:\temp\restore_dir\`. Likewise, the value 104 was inserted after the second backup operation and therefore wasn't present in the restored database in `c:\temp\restore_dir2\`.
24. Once your point-in-time recovery is complete, shut down the database and restart the database server without any of the backup-related options (for example, -a, -ad, and -ar).

Related Information

Point-in-time Recovery [page 962]
Database Recovery [page 960]
Tutorial: Restoring the Database to a Timestamp [page 965]
-ad Database Option [page 545]
-ru Database Option [page 557]
-ruo Database Option [page 558]

1.6.1.2.7 Restoring from an Image Backup

Restore a database that does not have incremental transaction logs to apply from an image backup.

Prerequisites

Create archive copies of the corrupt or damaged database file and transaction log before starting the restore process, to ensure you can start over if something goes wrong during recovery.

Procedure

1. Copy the database files back to their original location. For example, run the following command:

```
copy c:\temp\backup\demo.db C:\Users\Public\Documents\SQL Anywhere
        17\Samples
copy c:\temp\backup\demo.log C:\Users\Public\Documents\SQL Anywhere
        17\Samples
```

2. Restart the database server.

Results

The database is restored.

Related Information

Database Recovery with Multiple Transaction Logs [page 979]
The Automatic Recovery Process [page 962]
Types of Backup [page 941]
Recovering Uncommitted Operations (SQL Central) [page 963]

1.6.1.2.8 Restoring a Database from an Archive Backup (SQL Central)

Use SQL Central to restore a database from an archive backup.

Prerequisites

Create archive copies of the corrupt or damaged database file and transaction log before starting the restore process, to ensure you can start over if something goes wrong during recovery.

This task assumes you do not have any incremental backups of the transaction log that need to be applied.

If you created the backup of a strongly encrypted database with free page elimination turned on, then you must specify the encryption key for the database when restoring it.

Procedure

1. Run the following command to start a personal database server:

   ```
   dbeng17 -n server-name
   ```

2. Start SQL Central and connect to the utility database. Complete the following fields in the *Connect* window:
   a. In the *User ID* field, type **DBA**.
   b. In the *Password* field, type **passwd**.
   c. In the *Database Name* field, type **utility_db**.
3. Click *OK*.
4. Click *Tools* > *SQL Anywhere 17* > *Restore Database*.
5. Follow the instructions in the wizard.

Results

The database is restored.

Related Information

Database Recovery with Multiple Transaction Logs [page 979]
The Utility Database (utility_db) [page 309]
The Automatic Recovery Process [page 962]
Types of Backup [page 941]

Recovering Uncommitted Operations (SQL Central) [page 963]

1.6.1.2.9 Restoring a Database from an Archive Backup (Interactive SQL)

Use Interactive SQL to restore a database from an archive backup.

Prerequisites

Create archive copies of the corrupt or damaged database file and transaction log before starting the restore process, to ensure you can start over if something goes wrong during recovery.

This task assumes you do not have any incremental backups of the transaction log that need to be applied.

If you created the backup of a strongly encrypted database with free page elimination turned on, you must specify the encryption key for the database when restoring it.

Procedure

1. Run the following command to start a personal database server:

   ```
   dbeng17 -n server-name
   ```

2. Start Interactive SQL and connect to the utility database. Complete the following fields:
   a. In the *User ID* field, type **DBA**.
   b. In the *Password* field, type **passwd**.
   c. In the *Database Name* field, type **utility_db**.
3. Click *OK*.
4. Execute a RESTORE DATABASE statement, specifying the archive root.

   At this time, you choose to restore an archived database to its original location (the default), or to a different computer with different device names by using the RENAME clause.

Results

The database is restored.

Example

The following statement restores a database from an archive backup in file `c:\backup\archive.1` to the database file `c:\newdb\newdb.db`. The transaction log name and location are specified in the database.

```
RESTORE DATABASE 'c:\\newdb\\newdb.db'
FROM 'c:\\backup\\archive';
```

The following statement restores a database from an archive backup in file `c:\backup\archive.1` to the database file `c:\newdb\newdb.db`. The transaction log name and location are specified in the database. The encryption key is specified for the database.

```
RESTORE DATABASE 'c:\\newdb\\newdb.db'
FROM 'c:\\backup\\archive'
KEY '3Km57y1z';
```

Related Information

The Utility Database (utility_db) [page 309]
The Automatic Recovery Process [page 962]
Types of Backup [page 941]
Recovering Uncommitted Operations (SQL Central) [page 963]
RESTORE DATABASE Statement
-gu Database Server Option [page 452]

1.6.1.2.10 Restarting a Database from a Live Backup

Restart a database from a live backup made to a separate computer from the primary computer that is running your production database.

Prerequisites

You must have SQL Anywhere installed on the secondary computer.

You must have the BACKUP DATABASE system privilege.

Procedure

1. Copy the full backup transaction log file and the live backup transaction log to a new directory where they can be applied to the backup copy of the database file.

2. Rename the current transaction log file whose name matches the expected transaction log file name if one exists.
3. Start the database server with the -ad option to apply the transaction logs in the new directory and bring the database up to date:

   ```
   dbsrv17 "database-name.db" -ad directory-name
   ```

   The database server shuts down automatically once the transaction log is applied.
4. Start the database server in the normal way, allowing user access. Any new activity is written to a new transaction log.
5. Perform a full backup of the database.

   ```
   dbbackup -c "connection-string" path
   ```
6. Start a live backup of the transaction log to the secondary computer.

   ```
   dbbackup -c "connection-string" -l path\filename.log
   ```

Results

The database is restarted.

Related Information

Live Backups (Continuous Backups) [page 944]
The Automatic Recovery Process [page 962]
Types of Backup [page 941]
Recovering Uncommitted Operations (SQL Central) [page 963]

1.6.1.2.11 Database Recovery with Multiple Transaction Logs

If you need to recover your database and you have multiple transaction logs, then you must apply the transaction log files to the backup copy of your database in the correct order.

Use any of the following methods to apply transaction logs in the correct order:

- Use the -a server option to apply each transaction log individually to the backup copy of the database. You can use the Transaction Log utility (dblog) to determine the order in which transaction log files were generated. The utility generates and displays the earliest log offset in the transaction log, which can be an effective method for determining the order in which to apply multiple log files.
- Use the -ad server option to specify the location of the transaction log files. The database server determines the correct order for applying the transaction logs to the backup copy of the database based on the transaction log offsets.

- Use the -ar server option to have the database server apply log files associated with the database that are located in the same directory as the transaction log. The transaction log location is obtained from the database. The database server determines the correct order for applying the transaction logs to the backup copy of the database based on the log offsets.
- Use the Log Translation utility (dbtran) to translate one or more transaction logs into a `.sql` file that can be applied to the backup copy of the database after determining the correct order.

In this section:

Related Information

-a Database Option [page 544]
-ad Database Option [page 545]
-ar Database Option [page 547]
Transaction Log Utility (dblog) [page 1231]

1.6.1.2.11.1 Recovering a Database with Multiple Transaction Logs by Using the -ad Server Option

Use the -ad server option to recover a database by applying all the transaction logs from a specified directory to the backup copy of a database.

Prerequisites

Create backup copies of the corrupt or damaged database file and transaction log before starting the recovery process, to ensure you can start over if something goes wrong during recovery.

Procedure

1. Copy the backup transaction log and current transaction log into a directory.

2. Start the database server and apply the transaction logs to a backup copy of a database:

   ```
   dbsrv17 backup-copy-of-database -ad directory
   ```

   The database server uses the transaction log offsets in the transaction logs to determine the correct order in which to apply the transaction log files.

Results

The database server applies the transaction logs to the backup copy of the database and then shuts down.

Example

Start the database server and apply the transaction logs in directory `c:\backuplogs` to a backup copy of a database called `backupdemo.db`:

```
dbsrv17 backupdemo.db -ad c:\backuplogs
```

Related Information

The Automatic Recovery Process [page 962]
Recovering a Database with Multiple Transaction Logs by Using the -a Server Option [page 981]
Recovering Uncommitted Operations (SQL Central) [page 963]
-ad Database Option [page 545]

1.6.1.2.11.2 Recovering a Database with Multiple Transaction Logs by Using the -a Server Option

Use the -a server option to apply a single transaction log file to the backup copy of a database, and then shut the database server down.

Prerequisites

Create archive copies of the corrupt or damaged database file and transaction log before starting the recovery process, to ensure you can start over if something goes wrong during recovery.

Context

If you have multiple transaction logs, then apply them one at a time in the correct order, from oldest to most recent.

Procedure

1. Start the database server and apply a backup transaction log to the backup copy of a database:

   ```
   dbsrv17 backup-database.db -a backup-transaction-log.log
   ```

   The database server applies the backup transaction log to the backup copy of the database and then shuts down.
2. Start the database server and apply the active transaction log to the backup copy of the database:

   ```
   dbsrv17 backup-database.db -a current-transaction-log.log
   ```

   The database server applies the current transaction log to the backup copy of the database and then shuts down.

Results

The database is restored and brought up to date by applying the backup transaction log, and the transaction log that was active at the time the database failed.

Example

1. Start the database server and apply a backup transaction log called `backupdemo.log` to the backup copy of a database called `backupdemo.db`:

   ```
   dbsrv17 backupdemo.db -a backupdemo.log
   ```

   The database server applies the backup transaction log to the backup copy of the database and then shuts down.
2. Start the database server and apply the current transaction log called *demo*`.log` to the backup copy of the database:

   ```
   dbsrv17 backupdemo.db -a demo.log
   ```

   The database server applies the current transaction log to the backup copy of the database and then shuts down.

Related Information

The Automatic Recovery Process [page 962]
Recovering a Database with Multiple Transaction Logs by Using the -ad Server Option [page 980]
Recovering Uncommitted Operations (SQL Central) [page 963]
-a Database Option [page 544]

1.6.1.2.12 Recover from Media Failure

Two types of failure can cause your database to become unusable: system failure and media failure.

A **system failure** occurs when the computer or operating system fails while there are partially completed transactions. This type of failure can occur when the computer is inappropriately turned off or restarted, when another application causes the operating system to fail, or because of a power failure. After a system failure occurs, the database server recovers automatically when you next start the database. The results of each transaction that was committed before the system error are intact. All changes by transactions that were not committed before the system failure are canceled.

A **media failure** is more serious, and occurs when either the device storing the database file becomes unusable, or the database file and/or transaction log become corrupt.

Full recovery from media failure depends on whether the failure affects the database file or transaction log. If your database file is unusable but your transaction log is usable, then you can recover all committed changes by using a previous full backup and applying the changes in the transaction log.

If the media failure affects the transaction log, and you are using a transaction log mirror, then you can recover all data. If you do not have a transaction log mirror, but have been backing up the transaction log (incremental backups), then you can recover all data up to the last transaction log backup. If you do not have a transaction log mirror, and have not been backing up your transaction log, then you cannot recover information entered between the last database checkpoint and the time of the media failure. If you have a live backup, then you can recover all completed transactions.

Here are some measures that you can put in place to protect your data against media failures:

- Implement a good backup plan. For example, a full backup periodically, with incremental backups in between full backups.
- Test your backup to make sure that you can use it for recovery.
- Back up to a different device than the one your database is located on.
- Use a transaction log mirror in configurations where loss of the transaction log can lead to loss of key information, or the breakdown of a replication system.
- Place the transaction log on a separate device than the one that your database is located on. Placing the transaction log on a separate device can also improve performance by eliminating the need for disk head movement between the transaction log and the main database file. However, *do not place the transaction log in another location on a network directory*. Reading and writing pages over a network results in poor performance and possible file corruption.
- Some computers with two or more hard drives have only one physical disk drive with several logical drives or partitions: if you want reliable protection against media failure, then make sure that you have a computer with at least two physical storage devices.

- Maintain a live backup.

In this section:

Related Information

Database Creation [page 277]
Types of Backup [page 941]
Live Backups (Continuous Backups) [page 944]
Changing the Location of a Transaction Log (SQL Central) [page 294]
Changing the Location of a Transaction Log (Command Line) [page 295]

1.6.1.2.12.1 Recovering from Media Failure on the Database File

Recover a database when the only file lost to media failure was the database file.

Prerequisites

You must have the BACKUP DATABASE and VALIDATE ANY OBJECT system privileges.

Procedure

1. Make an extra backup copy of the current transaction log. Since the database file is unavailable, the transaction log contains the only record of the changes that have been made since the last backup.
2. Create a recovery directory to hold the files that you use during recovery.
3. Copy the database file from the last full backup to the recovery directory.
4. Apply the transactions held in the backed up transaction logs to the recovery database. Use one of the following methods:

Option	Action
Manually apply each transaction log in chronological order	1. Copy the transaction log file into the recovery directory. 2. Start the database server with the apply transaction log (-a) option: `dbsrv17 database-name.db -a log-name.log` The database server shuts down automatically once the transactions are applied. 3. Once you have applied all the backed up transaction logs, copy the online transaction log into the recovery directory. Apply the transactions from the online transaction log to the recovery database. `dbsrv17 database-name.db -a log-name.log`
Have the database server determine the correct order of the transaction logs and apply them automatically	1. Copy the offline and online transaction log files into the recovery directory. 2. Start the database server with the -ad option to specify the location of the transaction logs. The database server determines the correct order in which to apply the transaction logs based on the log offsets: `dbsrv17 database-name.db -ad log-directory` The database server shuts down automatically once the transactions are applied.

5. Perform validity checks on the database you plan to use for the recovery.
6. Make a backup.
7. Move the database file to the production directory.
8. Notify users that they can access the production database.

Results

The database is recovered.

Related Information

The Automatic Recovery Process [page 962]
Validating a Database (SQL Central) [page 990]
Recovering Uncommitted Operations (SQL Central) [page 963]

1.6.1.2.12.2 Recovering from Media Failure on a Transaction Log Mirror

Use the transaction log mirror to recover from media failure.

Context

If your database is a consolidated database in a SQL Remote installation, then you should use a transaction log mirror or a hardware equivalent.

Procedure

1. Make an extra copy of the backup of your database file taken at the time that the transaction log was started.
2. Identify which of the two files is corrupt. Run the Log Translation utility (dbtran) on the transaction log and on its mirror. The file that generates an error message is corrupt.

   The following command translates a transaction log named *demo*`.log`, placing the translated output into *demo*`.sql`:

   ```
   dbtran demo.log
   ```
3. Copy the correct file over the corrupt file.
4. Restart the database server.

Results

The database is recovered.

Related Information

Transaction Log Mirrors [page 301]
The Automatic Recovery Process [page 962]
Recovering Uncommitted Operations (SQL Central) [page 963]

1.6.1.2.12.3 Recovering from Media Failure on an Unmirrored Transaction Log

Recover your database partially from media failure by using an unmirrored transaction log.

Context

If your database is a consolidated database in a MobiLink or SQL Remote installation, then you should use a transaction log mirror or hardware equivalent.

> ⚠ Caution
>
> The dbeng17 command should only be used when the database is not participating in a MobiLink or SQL Remote system. If your database is a consolidated database in a SQL Remote replication system, then you may have to re-extract the remote databases.

Procedure

1. Make an extra backup copy of the database file. Without a transaction log, the database file contains the only record of the changes made since the last backup and the most recent checkpoint.
2. Delete or rename the transaction log file.
3. Restart the database with the -f option.

   ```
   dbsrv17 "database-name.db" -f
   ```

   Without the -f option, the database server reports the lack of a transaction log as an error. With the -f option, the database server restores the database to the most recent checkpoint and then rolls back any transactions that were not committed at the time of the checkpoint. A new transaction log is then created.

Results

The database is partially recovered. Any transactions not committed at the time of the most recent checkpoint are rolled back.

Related Information

Transaction Log Mirrors [page 301]
The Transaction Log [page 292]
The Automatic Recovery Process [page 962]

Recovering Uncommitted Operations (SQL Central) [page 963]
-f Database Option [page 552]

1.6.1.3 Timelines

Timelines establish a sequence to the changes made to a database.

Timelines are useful during complex database restoration when you need to identify the database file and set of log files to apply in sequence to restore the database. Proper use of timelines guarantee that there will not be any overlapping offsets across the transaction logs.

Timelines are present in the output of the Translation Log utility (dblog) and the Log Translation utility (dbtran). The format of a timeline is "`GUID timestamp`" where `GUID` is a unique identifier for the timeline and `timestamp` is the time when the timeline was created, in UTC.

There are three notable times when timelines are generated:

- when the database is created
- when performing unload/reload of the database
- when performing a point-in-time recovery on the database

When a timeline is generated by a point-in-time recovery, the database server remembers the original timeline for the database that is being restored and may use the original timeline information again in further point in time recoveries.

Related Information

Point-in-time Recovery [page 962]
Rebuilding Databases Involved in Synchronization or Replication (Manual)
Transaction Log Utility (dblog) [page 1231]

1.6.2 Database Validation

Database file corruption may not be reported until the database server tries to access the affected part of the database. Periodically check that your database is valid.

Depending on the options you specify, validation can include checksums, correctness of index data, and whether all table pages belong to objects in the database. You can also run an express validation that disables referential integrity checking to significantly improve performance.

Database Validation Recommendations

Ensure that no changes are made to the database while it is being validated. For example, start the database in read-only mode using the -r database option, or use the in-memory validation mode with the -im database server option.

If validation fails, then you should treat the situation as a media failure and recover the database from the previous backup.

Table Validation Recommendations

Validate the tables using the default options--this is the fastest method to validate tables. If you receive an error indicating that a table is corrupt, then rerun the validation with either exclusive data locks or with snapshot isolation. If this validation fails, then treat the situation as a media failure and recover the database from the previous backup.

Validation does not acquire exclusive access to the table being validated, so it may incorrectly report corruption if the table is modified during the validation. If you receive a validation error indicating that a table is corrupt, then rerun the validation with either exclusive data locks or snapshot isolation. For example, run the VALIDATE statement with the WITH DATA LOCK or WITH SNAPSHOT clause. If this validation succeeds, then the original corruption errors were false. If this validation fails, then treat the situation as a media failure and recover from the previous backup.

Database Backup Validation Recommendations

To validate a database backup, first start the database backup using the in-memory validation mode, and then run validation. In-memory validation mode prevents writes, and verifies that transactions can be recovered. In-memory validation mode can handle databases that have pending transactions, unlike the -r database server option.. For example, use the in-memory validation mode with the -im database server option.

Index Validation Recommendations

If an index is corrupt, then you may want to unload the database without indexes, and then reload the database. You can also use the REBUILD clause of the ALTER INDEX statement to correct index corruption.

- If errors are reported, you can drop all the indexes and keys on a table and recreate them. Any foreign keys to the table also need to be recreated.
- If you suspect a particular index, you can execute an ALTER INDEX...REBUILD statement to rebuild the corrupted index.
- Another solution for errors reported by the VALIDATE TABLE statement is to unload and reload your entire database. You should use the dbunload -u option so that the unload process does not try to use a possibly corrupt index to order the data.

In this section:

Related Information

Transaction Log Validation [page 294]
Text Index Concepts and Reference
Validating a Table (SQL Central) [page 996]
VALIDATE Statement
sa_validate System Procedure
Validation Utility (dbvalid) [page 1257]
ALTER INDEX Statement
-r Database Server Option [page 488]

1.6.2.1 Validating a Database (SQL Central)

Validate a database using SQL Central.

Prerequisites

You must have the VALIDATE ANY OBJECT system privilege.

Procedure

1. Ensure that no changes are made to the database while it is being validated. For example, start the database in read-only mode using the -r database option, or use the in-memory validation mode with the -im database server option.
2. Use the *SQL Anywhere 17* plug-in to connect to the database.
3. In the left pane, select the database.

4. From the *File* menu, click *Validate Database*.
5. Follow the instructions in the *Validate Database Wizard*.

Results

The database is validated.

If the database validation fails for a reason other than table corruption, then you should treat the situation as a media failure and recover the database from a previous backup.

If you receive a a validation error indicating that a table is corrupt, then rerun the *Validate Database Wizard* and choose either the *Use exclusive data locks* option or the *Use snapshot isolation* option. If this second validation succeeds, then the original corruption errors were false. If this validation fails, then you should treat the situation as a media failure and recover from the previous backup

Related Information

Validation Utility (dbvalid) [page 1257]
VALIDATE Statement
-r Database Server Option [page 488]

1.6.2.2 Validating a Database (SQL)

Validate a database, including database backups, using the VALIDATE statement or the sa_validate system procedure.

Prerequisites

You must have the VALIDATE ANY OBJECT system privilege.

Procedure

1. Start the database so that concurrent transactions cannot occur during the validation. Choose one of the following options:

Option	Action
To validate a database	Start the database in read-only mode using the -r database option, or use the in-memory validation mode with the -im database server option.
To validate a database backup	Start the database backup using the in-memory validation mode. In-memory validation mode prevents writes, verifies that transactions can be recovered, and can handle databases that have pending transactions..

2. Connect to the database, and choose one of the following options.

Option	Action
sa_validate stored procedure	Run the sa_validate stored procedure: `CALL sa_validate;` This stored procedure returns a single column, named Messages. If all tables are valid, the column contains `No errors detected`.
VALIDATE DATABASE statement	Execute the VALIDATE DATABASE statement.

Results

The database is validated.

If the database validation fails for a reason other than table corruption, then you should treat the situation as a media failure and recover the database from a previous backup.

If you receive a a validation error indicating that a table is corrupt, then rerun the validation with either exclusive data locks or snapshot isolation. For example specify the VALIDATE statement with the USE DATA LOCK claus or the USE SNAPSHOT clause. If this second validation succeeds, then the original corruption errors were false. If this validation fails, then you should treat the situation as a media failure and recover from the previous backup

Related Information

Validation Utility (dbvalid) [page 1257]
VALIDATE Statement
-r Database Server Option [page 488]
sa_validate System Procedure

1.6.2.3 Validating a Database (dbvalid)

Validate a database using the Validate database utility (dbvalid).

Prerequisites

You must have the VALIDATE ANY OBJECT system privilege.

Procedure

Run the dbvalid utility and ensure that no changes are made the database while it is being validated by specifying either the -ws or -wl option. The -ws option applies snapshot isolation to the tables, and the -wl option to applies exclusive locks to the tables. For example:

```
dbvalid -c "connection-string-ws"
```

```
dbvalid -c "connection-string-wl"
```

Results

The database is validated.

If validation fails, then you should treat the situation as a media failure and recover the database from the previous backup.

Related Information

Validation Utility (dbvalid) [page 1257]
VALIDATE Statement
-r Database Server Option [page 488]

1.6.2.4 Corruption Detection Using Checksums

Checksums are used to determine whether a database page has been modified on disk.

Two types of checksums are supported: global checksums and write checksums.

Global Checksums

Global checksums are enabled when a database is created. When you create a database with global checksums enabled, a checksum is calculated for each page just before it is written to disk. The next time the page is read from disk, the page's checksum is recalculated and compared to the checksum stored on the page. If the checksums are different, then the page has been modified on disk and an error occurs.

You can check whether a database was created with global checksums enabled by executing the following statement:

```
SELECT DB_PROPERTY ( 'Checksum' );
```

This query returns ON if global checksums are turned on; otherwise, it returns OFF.

The CHECKSUM clause of the ALTER DATABASE statement lets you disable global checksums for an existing database. Disabling checksums is not recommended. Once global checksums are disabled, they cannot be enabled. You can either use the -wc+ option to enable write checksums for the database, or you can rebuild the database with checksums enabled.

Write Checksums

By default, databases created with version 10 and 11 of SQL Anywhere do not have global checksums enabled. If you start a database created with SQL Anywhere 11 on a version 12 or later database server, then by default the database server creates write checksums, which are checksums that are added to pages only when they are written to disk.

Version 12 and later databases have global checksums enabled by default. If you create a new database and disable global checksums for the database, you can enable write checksums by using the -wc+ option or the START DATABASE statement.

You can check whether a database was started with write checksums by executing the following statement:

```
SELECT DB_PROPERTY ( 'WriteChecksum' );
```

This query returns ON if checksums are enabled only when pages are written to disk (because global checksums or write checksums are enabled); otherwise, it returns OFF.

Automatic Write Checksum Creation

In the following situations, write checksums are enabled for the database, regardless of the global checksum setting that was specified when the database was created:

Critical pages

The database server calculates write checksums for critical database pages in all databases, regardless of whether global checksums are enabled. These checksums are used to detect offline corruption, which can help reduce the chances of other data being corrupted as the result of a bad critical page. Because the database server calculates these checksums, if a database becomes corrupt that does not have checksums enabled, the database server shuts down with a fatal error.

As well, if you validate a database that does not have global checksums enabled, but that has a bad critical page, dbvalid can still return warnings about checksum violations.

Databases running on some storage media

When the database is running on storage media that may be less reliable, such as network or removable drives, the database server automatically enables write checksums for the database. Write checksums remain enabled as long as the database resides on such a device, and the pages are checksummed when they are written. If the database is moved to a more reliable storage device, the database server verifies the checksum for checksummed pages when they are brought into the database server cache.

Validating Checksums

You can validate disk pages for databases that use global or write checksums. If a database has global checksums enabled, then all database pages are validated. If a database has used only write checksums, then only pages with checksums are validated.

For databases with checksums enabled, a checksum is calculated for each database page and this value is stored when the page is written to disk. You can use the Validation utility (dbvalid), the VALIDATE statement, the sa_validate system procedure, or the *Validate Database Wizard* in SQL Central to perform checksum validation, which consists of reading the database pages from disk and calculating the checksum for the page. If the calculated checksum does not match the stored checksum for a page, the page has been modified or corrupted while on disk or while writing to the page. If one or more pages has been corrupted, an error is returned and information about the invalid pages appears in the database server messages window.

In this section:

Related Information

-wc Database Server Option [page 522]
-wc Database Option [page 561]
START DATABASE Statement
CREATE DATABASE Statement
ALTER DATABASE Statement
CREATE DATABASE Statement
VALIDATE Statement
Validation Utility (dbvalid) [page 1257]
sa_validate System Procedure

1.6.2.4.1 Validating a Table (SQL Central)

Validate a table using SQL Central.

Prerequisites

You must have the VALIDATE ANY OBJECT system privilege. It is best to validate when there is no other activity on the table.

Procedure

1. Use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, double-click *Tables*.

Option	Action
Validate a specific table.	1. In the left pane, double-click *Tables*. 2. Right-click the table and click *Validate*. 3. Click *OK*.
Validate all tables and materialized views	1. In the left pane, select the database. 2. From the *File* menu, click *Validate Database*. 3. Select the database to validate and click *Next*. 4. Enable the *Validate tables and materialized views* option and follow the instructions in the wizard.

Results

The table is validated.

If this validation fails, then treat the situation as a media failure and recover from the previous backup.

Related Information

VALIDATE Statement
ALTER INDEX Statement
Validation Utility (dbvalid) [page 1257]

1.6.3 Task Automation Using Schedules and Events

Automate database administration tasks with schedules and events

Automate Tasks

Automate routine tasks by adding an **event** to a database, and providing a schedule for the event. Whenever one of the times in the schedule passes, the database server runs a sequence of actions called an **event handler**.

To schedule back ups, use a maintenance plan..

Trigger Tasks

Database administration also requires taking action when certain conditions occur. For example, it may be appropriate to email a notification to a system administrator when a disk containing the transaction log is filling up so that the administrator can handle the situation. These tasks too can be automated by defining event handlers for one of a set of **system events**.

In this section:

Related Information

How the Database Server Checks for Scheduled Events [page 1007]
How Event Handlers Are Executed [page 1008]
Trigger Conditions for Events [page 1003]
Debugging an Event Handler [page 1010]
CREATE EVENT Statement
EVENT_PARAMETER Function [System]

1.6.3.1 Events

You can automate routine tasks by adding an event to a database, and providing a schedule for the event.

The following types of events are supported:

Scheduled events

Scheduled events have an associated schedule and execute at specified times.

System events

System events are associated with a particular type of condition that is tracked by the database server.

Manual events

Manual events are fired explicitly using the TRIGGER EVENT statement.

User trace events

User trace events are used to log information about an application to an event tracing session. These events are visible to all connections to a database.

After each execution of an event handler, a COMMIT occurs if no errors occurred. A ROLLBACK occurs if there was an error.

Related Information

Schedules [page 998]
System Events [page 1001]
Event Tracing [page 1012]
Triggering an Event Handler [page 1009]
CREATE TEMPORARY TRACE EVENT Statement

1.6.3.2 Schedules

By scheduling activities you can ensure that a set of actions is executed at a set of preset times.

The scheduling information and the event handler are both stored in the database itself.

Although this is not usually necessary, you can define complex schedules by associating more than one schedule with a named event. For example, a retail outlet might want an event to occur once per hour during hours of operation, where the hours of operation vary based on the day of the week. You can achieve the same effect by defining multiple events, each with its own schedule, and by calling a common stored procedure.

When scheduling events, you can use either full-length English day names (Monday, Tuesday, and so on) or the abbreviated forms of the day (Mon, Tue, and so on). You must use the full-length English day names if you want the day names to be recognized by a server running in a language other than English.

The following examples give some ideas for scheduled actions that may be useful.

Example

Perform an incremental backup daily at 1:00 A.M.:

```
CREATE EVENT IncrementalBackup
SCHEDULE
 START TIME '1:00 AM' EVERY 24 HOURS
HANDLER
BEGIN
 BACKUP DATABASE DIRECTORY 'c:\\backup'
 TRANSACTION LOG ONLY
 TRANSACTION LOG RENAME MATCH
END;
```

Summarize orders at the end of each business day.

```
CREATE EVENT Summarize
SCHEDULE
 START TIME '6:00 pm'
 ON ( 'Monday', 'Tuesday', 'Wednesday', 'Thursday',
  'Friday' )
HANDLER
 BEGIN
 INSERT INTO OrderSummary
  SELECT CURRENT DATE,
     COUNT( * ),
     SUM( amount )
  FROM Orders
  WHERE date_ordered = current date
END;
```

In this section:

Related Information

CREATE EVENT Statement

1.6.3.2.1 Schedule Definition

To permit flexibility, schedule definitions have several components to them.

Name

Each schedule definition has a name. You can assign more than one schedule to a particular event, which can be useful in designing complex schedules.

Start time

You can define a start time for the event, which is the time when execution begins.

Range

As an alternative to a start time, you can specify a range of times for which the event is active. The event occurs between the start and end time specified. Frequency is determined by the specified recurrence.

Recurrence

Each schedule can recur. The event is triggered on a frequency that can be given in hours, minutes, or seconds on a set of days that can be specified as days of the week or days of the month. Recurring events include an EVERY or ON clause.

In this section:

1.6.3.2.1.1 Creating a Schedule for an Event

Define a schedule for an event to ensure that an action is executed at a preset time.

Prerequisites

You must have either the ALTER ANY OBJECT or MANAGE ANY EVENT system privilege.

Procedure

1. Use the *SQL Anywhere 17* plug-in to connect to your database.
2. In the left pane, double-click *Events*.
3. In the right pane, double-click the event for which you want to create a schedule.
4. Click the *Schedules* tab.
5. Click *File* > *New* > *Schedule*.
6. Follow the instructions in the *Create Schedule Wizard*.

Results

A schedule is created for the event.

Related Information

ALTER EVENT Statement

1.6.3.3 System Events

Each system event provides a hook on which you can program a set of actions.

The database server tracks the events for you, and executes the actions (as defined in the event handler) when the system event satisfies a provided **trigger condition**.

By defining event handlers to execute when a chosen system event type occurs and satisfies a trigger condition that you define, you can improve the security and safety of your data, and help ease administration. The actions of an event handler are committed if no error is detected during execution, and rolled back if errors are detected.

In this section:

1.6.3.3.1 System Event Types

There are several system event types.

Audit events

A subset of system trace events that log information to the database's audit log.

BackupEnd

You can use the BackupEnd event type to take action at the end of a backup.

Connection events

When a connection is made (Connect) or when a connection attempt fails (ConnectFailed). You may want to use these events for security purposes. As an alternative to a connect event handler, you may want to consider using a login procedure.

DatabaseStart

You can use the DatabaseStart event type to take action when a database is started.

Deadlock

You can use the Deadlock event to take action when a deadlock occurs. The event handler can use the sa_report_deadlocks procedure to obtain information about the conditions that led to the deadlock. When using the Deadlock event, you should configure the database server to capture deadlock information by setting the log_deadlocks option to On, and by enabling the RememberLastStatement feature using sa_server_option or the -zl server option.

Deadlock events fire for connection deadlocks and thread deadlocks. A deadlock event provides no information beyond what is available via the sa_report_deadlocks system procedure. However, using this event allows you to act on the deadlock in a timely manner. A quick response may be important since the amount of deadlock-related information the database server maintains is limited.

Disconnect

You can use the Disconnect event to take action when a user or application disconnects.

Free disk space

Tracks the available disk space on the device holding the database file (DBDiskSpace), the transaction log file (LogDiskSpace), or temporary file (TempDiskSpace).

You may want to use disk space events to alert administrators of a disk space shortage.

You can specify the -fc option when starting the database server to implement a callback function when the database server encounters a file system full condition.

File size

The file reaches a specified size. This can be used for the database file (GrowDB), the transaction log (GrowLog), or the temporary file (GrowTemp).

You may want to use file size events to track unusual actions on the database, or monitor bulk operations.

GlobalAutoincrement

When the number of remaining values for a column defined with GLOBAL AUTOINCREMENT is less than one percent of its range, the GlobalAutoincrement event fires. This can be used to request a new value for the global_database_id option based on the table and number of remaining values that are supplied as parameters to this event. To get the remaining values for the table within the event, use the EVENT_PARAMETER function with the RemainingValues and TableName parameters. RemainingValues returns the number of remaining values that can be generated for the column, while TableName returns the table containing the GLOBAL AUTOINCREMENT column that is near the end of its range.

RAISERROR error

When a RAISERROR statement is executed, you can use the RAISERROR event type to take actions. The error number used in the RAISERROR statement can be determined within the event handler using the EVENT_CONDITION function (for example, `EVENT_CONDITION( 'ErrorNumber' )`).

Idle time

The database server has been idle for a specified time (ServerIdle). You may want to use this event type to perform routine maintenance operations at quiet times.

Database mirroring system event types

The following system events are supported for database mirroring:

MirrorFailover

This event fires each time a database server takes ownership of the mirrored database. For example, it fires when a server first starts and determines that it should own the database. It also fires when a server previously acting as the mirror determines that the primary server has gone down and, after consulting with the arbiter, determines that it should take ownership.

MirrorServerDisconnect

When the connection between the primary server and mirror server or the connection between the primary server and arbiter server is lost, the MirrorServerDisconnect event fires. Within the handler for this event, the value of EVENT_PARAMETER('MirrorServerName') is the name of the server whose connection was lost. This event only fires on the primary server.

Related Information

Deadlocks

login_procedure Option [page 755]
sa_report_deadlocks System Procedure
log_deadlocks Option [page 751]
-fc Database Server Option [page 425]
EVENT_PARAMETER Function [System]

1.6.3.3.2 Trigger Conditions for Events

Each event definition has a system event associated with it, and one or more trigger conditions.

The event handler is triggered when the trigger conditions for the system event are satisfied.

The trigger conditions are included in the WHERE clause of the CREATE EVENT statement, and can be combined using the AND keyword. Each trigger condition is of the following form:

```
event_condition( condition-name ) comparison-operator value
```

The `condition-name` argument is one of a set of preset strings, which are appropriate for different event types. For example, you can use DBSize (the database file size in megabytes) to build a trigger condition suitable for the GrowDB system event. The database server does not check that the condition-name matches the event type: it is your responsibility to ensure that the condition is meaningful in the context of the event type.

Example

- Limit the transaction log size to 10 MB:

```
CREATE EVENT LogLimit
TYPE GrowLog
WHERE event_condition( 'LogSize' ) > 10
HANDLER
BEGIN
  IF EVENT_PARAMETER( 'NumActive' ) = 1 THEN
   BACKUP DATABASE
   DIRECTORY 'c:\\logs'
   TRANSACTION LOG ONLY
   TRANSACTION LOG RENAME MATCH;
  END IF;
END;
```

- Notify an administrator when free disk space on the device containing the database file falls below 10%, but do not execute the handler more than once every five minutes (300 seconds):

```
CREATE EVENT LowDBSpace
TYPE DBDiskSpace
WHERE event_condition( 'DBFreePercent' ) < 10
AND event_condition( 'Interval' ) >= 300
HANDLER
BEGIN
 CALL xp_sendmail( recipient='DBAdmin',
    subject='Low disk space',
    "message"='Database free disk space '
```

```
    || EVENT_PARAMETER( 'DBFreeSpace' ) );
END;
```

- Notify an administrator of a possible attempt to break into the database:

```
CREATE EVENT SecurityCheck
TYPE ConnectFailed
HANDLER
BEGIN
 DECLARE num_failures INT;
 DECLARE mins INT;
 INSERT INTO FailedConnections( log_time )
 VALUES ( CURRENT TIMESTAMP );
 SELECT COUNT( * ) INTO num_failures
 FROM FailedConnections
 WHERE log_time >= DATEADD( minute, -5,
  current timestamp );
 IF( num_failures >= 3 ) THEN
  SELECT DATEDIFF( minute, last_notification,
   current timestamp ) INTO mins
  FROM Notification;
  IF( mins > 30 ) THEN
   UPDATE Notification
   SET last_notification = current timestamp;
   CALL xp_sendmail( recipient='DBAdmin',
      subject='Security Check', "message"=
       'over 3 failed connections in last 5 minutes' )
  END IF
 END IF
END;
```

- Run a process when the server has been idle for ten minutes. Do not execute more frequently than once per hour:

```
CREATE EVENT Soak
TYPE ServerIdle
WHERE event_condition( 'IdleTime' ) >= 600
AND event_condition( 'Interval' ) >= 3600
HANDLER
BEGIN
 MESSAGE ' Insert your code here ... '
END;
```

1.6.3.4 Event Handlers

Event handlers execute on a separate connection from the action that triggered the event, and do not interact with client applications.

They execute with the privileges of the creator of the event.

Event handlers, whether for scheduled events or for system event handling, contain compound statements, and are similar in many ways to stored procedures. You can add loops, conditional execution, and so on.

You can also use the SQL Anywhere debugger to debug event handlers.

After each execution of an event handler, a COMMIT occurs if no errors occurred. A ROLLBACK occurs if there was an error.

Context Information for Event Handlers

Unlike stored procedures, event handlers do not take any arguments. You can use the EVENT_PARAMETER function to access information about the context in which an event was triggered. The information returned includes the connection ID and user ID that caused an event to be triggered, and the event name and the number of times it has been executed.

Test Event Handlers

During development, you want event handlers to be triggered at convenient times. You can use the TRIGGER EVENT statement to explicitly cause an event to execute, even when the trigger condition or scheduled time has not occurred. However, TRIGGER EVENT does not cause disabled event handlers to be executed.

While it is not good practice to develop event handlers on a production database, you can disable event handlers explicitly using the ALTER EVENT statement.

Code Sharing

It can be useful to use a single set of actions to handle multiple events. For example, you may want to take a notification action if disk space is limited on any of the devices holding the database or log files. To do this, create a stored procedure and call it in the body of each event handler, passing any needed context information as parameters to the procedure.

Debug Event Handlers

Debugging event handlers is very similar to debugging stored procedures. The event handlers appear in the events list.

Hide Event Handlers

You can use the ALTER EVENT statement with the SET HIDDEN clause to hide the definition of an event handler. Specifying the SET HIDDEN clause results in the permanent obfuscation of the event handler definition stored in the action column of the ISYSEVENT system table.

Limit Active Events

You can also determine how many instances of a particular event handler are currently active using EVENT_PARAMETER function with the NumActive context name. This function is useful to limit an event handler so that only one instance executes at any given time.

In this section:

Related Information

EVENT_PARAMETER Function [System]
TRIGGER EVENT Statement

1.6.3.4.1 How the Database Server Checks for System Events

System events are classified according to their **event type**, as specified in the CREATE EVENT statement.

There are two kinds of event types:

Active event types

Some event types are the result of action by the database server itself. These active event types include growing database files, or the start and end of different database actions (BackupEnd and so on) or RAISERROR.

When the database server takes the action, it checks to see whether the trigger conditions defined in the WHERE clause are satisfied, and if so, triggers any events defined for that event type.

Polled event types

Some event types, such as free disk space types (DBDiskSpace and so on) and IdleTime type, are not triggered solely by database actions.

For these types of events, the database server polls every thirty seconds, starting approximately thirty seconds after the database server is started.

For the IdleTime event type, the database server checks whether the server has been idle for the entire thirty seconds. If no requests have started and none are currently active, it adds the idle check interval time in seconds to the idle time total; otherwise, the idle time total is reset to 0. The value for IdleTime is therefore always a multiple of thirty seconds. When IdleTime is greater than the interval specified in the trigger condition, event handlers associated with IdleTime are fired.

1.6.3.4.2 How the Database Server Checks for Scheduled Events

The calculation of scheduled event times is done when the database server starts, and each time a scheduled event handler completes.

The calculation of the next scheduled time is based on the increment specified in the schedule definition, with the increment being added to the previous start time. If the event handler takes longer to execute than the specified increment, so that the next time is earlier than the current time, the database server increments until the next scheduled time is in the future.

For example, an event handler that takes sixty-five minutes to execute and is requested to run every hour between 9:00 and 5:00 will run every two hours, at 9:00, 11:00, 1:00, and so on.

To run a process such that it operates between 9:00 and 5:00 and delays for some period before the next execution, you could define a handler to loop until its completion time has passed, with a WAITFOR statement between each iteration.

If you are running a database server intermittently, and it is not running at a scheduled time, the event handler does not run at startup. Instead, the next scheduled time is computed at startup. If, for example, you schedule a backup to take place every night at one o'clock, but regularly shut down the database server at the end of each work day, the backup never takes place.

If the next scheduled execution of an event is more than one hour away, the database server will recalculate its next scheduled time on an hourly basis. This allows events to fire when expected when the system clock is adjusted because of a change to or from Daylight Savings Time.

1.6.3.4.3 How Event Handlers Are Executed

When an event handler is triggered, a temporary internal connection is made on which the event handler is executed.

The handler is not executed on the connection that caused the handler to be triggered, so statements such as MESSAGE...TO CLIENT, which interact with the client application, are not meaningful within event handlers. Similarly, statements that return result sets are not permitted.

The temporary connection on which the handler is executed does not count towards the connection limit for licensing purposes, and the procedure specified by the login_procedure option is not executed for event connections.

Event handlers execute on a separate connection, with the privileges of the event owner. You can also call a procedure from within the event handler, in which case the procedure executes with the privileges of procedure owner. The separate connection for the event handler does not count towards the ten-connection limit of the personal database server.

Any event errors are logged to the database server message log.

> **i** Note
>
> The transaction in an event handler is committed if no errors are detected during execution, and rolled back if errors are detected.

Related Information

Exception Handling and Atomic Compound Statements

1.6.3.4.4 Creating a Database Event

Create an event to automate a routine task.

Prerequisites

You must have either the MANAGE ANY EVENT or CREATE ANY OBJECT system privilege.

Context

You can create scheduled events, system events, and manual events.

If you are developing event handlers, you can add schedules or system events to control the triggering of an event later, either using SQL Central or the ALTER EVENT statement.

If you create an event handler without a schedule or system event to trigger it, it is executed only when manually triggered.

Procedure

1. Use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, right-click *Events* and click *New* > *Event*.
3. Follow the instructions in the *Create Event Wizard*.

Results

The event is added to the database.

Related Information

Task Automation Using Schedules and Events [page 997]
Triggering an Event Handler [page 1009]
CREATE EVENT Statement
ALTER EVENT Statement

1.6.3.4.5 Triggering an Event Handler

Trigger an event.

Prerequisites

You must have the MANAGE ANY EVENT system privilege.

Context

Any event handler can be triggered manually in addition to those occasions when it executes because of a schedule or system event.

Triggering events manually can be useful during the development of event handlers, and also, for certain events, in production environments. For example, if you have a monthly sales report scheduled, you might want to obtain a sales report for a reason other than the end of the month.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, double-click *Events*.
3. Right-click the event and click *Trigger*.

   The event must be enabled before you can trigger it. To enable an event, right-click it and click *Enabled*.
4. In the *Parameters* field, type a comma-separated list of parameters for the event. For example:

   ```
   parameter=value,parameter=value
   ```
5. Click *OK*.

Results

The event handler is triggered.

Related Information

TRIGGER EVENT Statement

1.6.3.4.6 Debugging an Event Handler

Debug an event handler.

Prerequisites

You must have the DEBUG ANY PROCEDURE system privilege.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. Click *Mode* > *Debug*.
3. In the left pane, double-click *Events*.
4. In the right pane, double-click the event you want to debug.
5. On the *SQL* tab in the right pane, press F9 to set a breakpoint.

Results

Trigger the event handler using the TRIGGER EVENT statement. The execution stops at the breakpoint you set.

Related Information

The SQL Anywhere Debugger
TRIGGER EVENT Statement

1.6.3.4.7 Hiding an Event Handler

Hide the definition for an event handler using the ALTER EVENT statement.

Prerequisites

You must have either the MANAGE ANY EVENT or ALTER ANY OBJECT system privilege.

Procedure

1. Connect to the database.
2. Execute an ALTER EVENT statement with the SET HIDDEN clause:

```
ALTER EVENT event-name SET HIDDEN
```

Results

The event handler is permanently obfuscated in the event handler definition that is stored in the action column of the ISYSEVENT system table.

Example

The following statement hides the definition of the event handler secret_event:

```
ALTER EVENT secret_event SET HIDDEN
```

Related Information

ALTER EVENT Statement
SYSEVENT System View

1.6.4 Event Tracing

Event tracing records information about system-defined and user-defined trace events to an event trace data (ETD) file.

A trace session is made up of trace events (specific points in the database server software or your SQL application) that collect information that is logged to a target. Targets are the location (such as a file) where the database server logs trace events.

The information that is logged during an event tracing session includes the information from the specified system- and user-defined trace events. Event tracing can be used to diagnose database server and application issues, including performance issues, on production databases.

System trace events

These are trace events that the database server generates. Trace events are generated for operations such as starting or ending a checkpoint and starting or stopping a database. You can query the available system events and system event fields by using the sp_trace_events and sp_trace_event_fields system procedures. Event fields contain pertinent information about the event. System events are not maintained when you rebuild a database, and the set of system events that is available depends on the version of the database server. System trace events can change across major releases.

Audit events are a subset of system trace events and log information to the database's audit log. Query the available audit events and audit event fields by using the sp_trace_events and sp_trace_event_fields system procedures. However, you must have the MANAGE AUDITING system privilege.

> **i** Note
>
> The audit events do not directly correspond to available values of the types parameter in the sa_enable_auditing type system procedure. When you enable auditing by using the

sa_enable_auditing_type system procedure, specify what type of information to audit. The actual auditing events are system events generated by the database server.

User trace events

These trace events log information from an application to an event tracing session. User trace events are visible to all connections to the database. You can create a user trace event by using the CREATE TEMPORARY TRACE EVENT statement.

Event tracing targets

The only supported event tracing target is an event trace data (ETD) file. The ETD file is platform independent and can be processed with the dbmanageetd utility. You can also retrieve data from an ETD file using the sp_read_etd system procedure.

Trace sessions

A trace session is a collection of trace events and targets for a database. Trace sessions are visible to all connections to the database, and trace sessions persist until the database is shut down.

Event severities

The following SQL severity levels are defined for all trace event types:

Level	Severity value range
ALWAYS	0
CRITICAL	1-50
ERROR	51-100
WARNING	101-150
INFORMATION	151-200
DEBUG	201-255

Setting up Event Tracing for a Database

Perform the following steps to set up event tracing for a database:

1. Create user trace events within your application for the information you want included in the diagnostic log.
2. Create a procedure that calls the NOTIFY TRACE EVENT statement.
3. Create a trace session and add trace events to the session.

4. Start the event tracing session.

In this section:

Related Information

Viewing the Contents of the Event Trace Data (ETD) Diagnostic Log File [page 1018]
sp_read_etd System Procedure
sp_trace_events System Procedure
sp_trace_event_fields System Procedure
CREATE TEMPORARY TRACE EVENT Statement
CREATE TEMPORARY TRACE EVENT SESSION Statement

1.6.4.1 Creating a User Trace Event

Create user trace events to track information within your application.

Prerequisites

You must have the MANAGE ANY TRACE SESSION and NOTIFY TRACE EVENT system privileges.

Procedure

1. Create the required user trace event for your database.

For example:

```
CREATE TEMPORARY TRACE EVENT my_event( id INTEGER, information LONG VARCHAR );
```

You can create a stored procedure that runs when the database starts to create all required user trace events.

2. Execute a NOTIFY TRACE EVENT statement for each user-defined event that is being traced for the database.

   For example:

   ```
   NOTIFY TRACE EVENT my_event( 1, 'Hello world' );
   ```

   The NOTIFY TRACE EVENT statement logs information about trace events to any trace session that is interested in the event. If an event is not defined when NOTIFY TRACE EVENT is executed, the database server generates an error. If an event does not exist, you can place the NOTIFY TRACE EVENT statement in a TRY block to define the behavior.

Results

The trace event exists until the database is shut down.

Next Steps

Create a trace session for the database.

Related Information

Creating a Trace Session [page 1016]
CREATE TEMPORARY TRACE EVENT Statement
NOTIFY TRACE EVENT Statement
TRY Statement

1.6.4.2 Creating a Trace Session

Create a trace session that can record event tracing information to the diagnostic log.

Prerequisites

You must have the MANAGE ANY TRACE SESSION system privilege.

Any user trace events that are referenced in NOTIFY TRACE EVENT statements for the database must exist, unless the NOTIFY TRACE EVENT statement is contained in a TRY block that handles any errors if the user trace event does not exist.

Procedure

Execute a CREATE TEMPORARY TRACE EVENT SESSION statement.

For example:

```
CREATE TEMPORARY TRACE EVENT SESSION my_session
    ADD TRACE EVENT my_event, -- user event
    ADD TRACE EVENT SYS_ConsoleLog_Information -- system event
    ADD TARGET FILE ( SET filename_prefix='my_trace_file' );
```

The ADD TRACE EVENT clauses specify user and system trace events that are part of the session. The ADD TARGET FILE clause specifies the name of the logging target and, optionally, any parameters associated with that target.

Results

The trace session is created for the database, but it must be started manually.

Next Steps

Start the trace session for the database.

Related Information

Creating a User Trace Event [page 1014]
CREATE TEMPORARY TRACE EVENT SESSION Statement

1.6.4.3 Starting or Stopping a Trace Session

Start a trace session so that event tracing information is recorded to the tracing targets, and stop a trace event session when you are finished recording information.

Prerequisites

You must have the MANAGE ANY TRACE SESSION system privilege.

Procedure

Execute an ALTER TRACE EVENT SESSION statement that specifies the required state of the trace session: START or STOP.

For example, the following statement starts the trace session named my_session:

```
ALTER TRACE EVENT SESSION my_session
     STATE = START;
```

The following statement stops the trace session named my_session:

```
ALTER TRACE EVENT SESSION my_session
     STATE = STOP;
```

Results

Once started, the session runs until it is explicitly stopped or until the database is shut down.

Next Steps

View the contents of the diagnostic log file by using the dbmanageetd utility.

Related Information

ALTER TRACE EVENT SESSION Statement
Event Trace Data (ETD) File Management Utility (dbmanageetd) [page 1157]

1.6.4.4 Viewing the Contents of the Event Trace Data (ETD) Diagnostic Log File

View the contents of the diagnostic log file using the event trace data (ETD) File Management utility (dbmanageetd).

Prerequisites

You must have logged event tracing information from a tracing session to an ETD file.

Procedure

1. Run the dbmanageetd utility to view the contents of the diagnostic log file in text format. For example:

   ```
   dbmanageetd diagnostic-log.etd -o diagnostic-log.txt
   ```

2. Open the file in a text editor.

   The following table shows the possible severity levels for tracing events.

Level	Severity value range
ALWAYS	0
CRITICAL	1-50
ERROR	51-100
WARNING	101-150
INFORMATION	151-200
DEBUG	201-255

Results

The diagnostic log is generated and can be viewed in a text editor.

Related Information

Event Trace Data (ETD) File Management Utility (dbmanageetd) [page 1157]

1.6.4.5 Listing Available Trace Events

View the user-defined and system-defined trace events, along with their fields, recorded in the diagnostic log.

Prerequisites

The MANAGE ANY TRACE SESSION system privilege is required.

Procedure

1. To view a list of all trace events, run the sp_trace_events system procedure.
2. To view a list of the fields for the trace events, run the sp_trace_event_fields system procedure.

   The following table shows the possible severity levels for tracing events.

Level	Severity value range
ALWAYS	0
CRITICAL	1-50
ERROR	51-100
WARNING	101-150
INFORMATION	151-200
DEBUG	201-255

Results

The database server returns a result set listing the supported trace events.

Related Information

sp_trace_events System Procedure
sp_trace_event_fields System Procedure

1.6.5 Troubleshooting Database Issues

There are many tools and methods you can use to troubleshoot database issues.

In this section:

Related Information

Tips for Improving Performance [page 1436]
SQL Anywhere Profiler [page 1483]
SQL Anywhere Monitor [page 1265]
Performance Improvements, Diagnostics, and Monitoring [page 1263]

1.6.5.1 Troubleshooting: Understanding Unexpected Changes in the Database Size

Your database file may increase or decrease in size unexpectedly.

Database pages are freed when all records on the page are deleted. When a database page is freed, it becomes available for reuse, but it cannot be removed from the file. However, future INSERT and UPDATE statements can use the freed pages.

Whenever modifications such as inserts, updates, or deletes are made to the database, entries are added to the rollback log, which is stored within the system dbspace. If many of these operations are performed before a commit is executed, the rollback log can become very large and may increase the size of the database.

The checkpoint log is stored at the end of the system dbspace. When the database server shuts down, the checkpoint log is truncated and the system dbspace shrinks. Pages that were freed by DELETE or TRUNCATE operations remain within the database file for future reuse and cannot be removed from the file.

If the size of your database file is increasing, or is not decreasing as expected:

- Execute COMMITs frequently if you are using INSERT, UPDATE, or DELETE statements. Pages allocated for the rollback log are freed for reuse in the system dbspace when a COMMIT is performed.
- Execute CHECKPOINTs occasionally when you are using UPDATE or DELETE statements, or if you are using INSERT statements and large indexes are involved. Pages in the checkpoint log become available for reuse by the checkpoint log after each checkpoint.

- Execute TRUNCATE TABLE, which can result in page-level deletes. In these cases, copies of the pages do not need to be added to the checkpoint log and individual row-level operations do not need to be added to the rollback log. Pages freed by TRUNCATE TABLE are only reusable after the next checkpoint. TRUNCATE TABLE results in page-level deletes when the following conditions are true:
  - There are no foreign keys to, or from, the table being truncated.
  - TRUNCATE TABLE is not being executed within a trigger.
  - TRUNCATE TABLE is not being executed with an atomic operation.
  - The checkpoint log pages are written to the end of the system dbspace file. These pages are removed when the database is shut down.

Rebuilding the database can decrease the size of the database because the rebuilt database has fewer free pages.

Related Information

Information Utility (dbinfo) [page 1165]

1.6.5.2 Troubleshooting: Reporting an Error

Submit performance data, such as crash reports and diagnostic information reports, to help improve the product.

Prerequisites

If necessary, configure your proxy host and port settings.

Context

Every effort is made to ensure that the report does not contain any sensitive information such as your network user name or password.

Error reports are stored on your computer until you delete them.

Procedure

Run the following command:

```
dbsupport -sa
```

Results

All crash report and diagnostic information stored in the diagnostic directory are submitted to the software development team.

In this section:

Related Information

Support Utility (dbsupport) [page 1221]
SQL Anywhere Performance and Diagnostic Tracking

1.6.5.2.1 Regularly Submit Performance Data

To help improve the product, the software has the ability to automatically report performance data to the software development team.

During installation, this feature is enabled by default. You can also enable this feature:

- In Interactive SQL, by clicking *Tools* > *Options* > *Support*
- In SQL Central, by clicking *Help* > *SQL Anywhere 17* > *Configure Update Checker*.
- With the Support utility (dbsupport), by specifying the -cc autosubmit option.

OEM Users

By default, when you deploy the administration tools, the feature that automatically submits performance data to the software development team is disabled. But, your users can enable or disable this feature within the administration tools. To prevent your users from changing whether their data is submitted or not, set the showPerformanceDataUI option to false in the `OEM.ini` file.

To enable your users to submit performance data to SAP, you must use the Support utility (dbsupport) with the -cc autosubmit option.

Related Information

Administration Tools Configuration

Support Utility (dbsupport) [page 1221]

1.6.5.3 Troubleshooting: Database Server Startup

There are several ways to troubleshoot the database server startup.

Ensure That Your Transaction Log File Is Valid

The database server won't start if the existing transaction log is invalid.

For example, during development you may replace a database file with a new version, without deleting the transaction log at the same time. However, doing so causes the transaction log file to be different than the database, and results in an invalid transaction log file.

Ensure That You Have Enough Disk Space For Your Temporary File

A temporary file is used to store information while the software is running.

This file is usually stored in the directory pointed to by the SATMP environment variable, typically `c:\temp`.

If you do not have enough disk space available to the temporary directory, you will have problems starting the server.

Debug Network Communications Startup Problems

If the server fails to start with a TCP/IP or a communication-related error, you can use the -z debugging server option.

The startup information appears in the database server messages window. You can also use the -o option to log the results to an output file.

Related Information

-z Database Server Option [page 532]
-o Database Server Option [page 471]
SATMP Environment Variable [page 578]

1.6.5.4 Troubleshooting: Database Server Performance Warnings

Database server performance warnings appear in the database server message log.

These warnings are reported when a database or database server reaches a state where its performance may be affected.

In this section:

1.6.5.4.1 Performance Warning: Server Cache Size Is Too Small for Database %1

This warning is reported if the database server's maximum cache size is less than 10% of the total size of all the open dbspaces for a database.

The cache size is the upper limit of the database server's buffer pool size, as computed by the server or as explicitly specified by the -ch option.

For example, a database server hosting a 21 GB database would issue this warning if the server's maximum cache size was less than 2.1 GB. This warning is computed on a per-database basis, and may not be issued even if multiple databases are started on the same database server.

Related Information

Tip: Use the Cache to Improve Performance [page 1440]
Dynamic Cache Sizing [page 1443]
Statistics That Monitor Cache Size [page 1445]
-ch Database Server Option [page 407]

1.6.5.4.2 Performance Warning: Database %3 Has a Page Size of %1 That Does Not Match Maximum Of %2 Set for Server, Causing Inefficient Use of Cache

The page size of the database server's buffer pool must be at least as large as the page size of any database started on that database server.

A database server can use a larger page size for the cache, which can be specified using the -gp option. However, in a SQL Anywhere database server, there must be a 1:1 mapping between database pages and page frames in the buffer pool. Each page frame in memory can contain only one database page, and any page may appear in only one page frame. If the cache uses 8 KB pages, but the database uses 4 KB pages, then each cache page frame in memory contains only 4 KB of data, and the other 4 KB of that page frame are wasted. This situation results in a cache size of half the desired amount.

Related Information

Maximum Page Size Considerations [page 333]
-gp Database Server Option [page 443]

1.6.5.4.3 Performance Warning: No Unique Index or Primary Key for Table %1 in Database %2

The performance warning is reported when a table is updated that contains more than 10 rows and that does not have a primary key or a unique index.

The transaction log contains logical row operations (UPDATE, INSERT, and DELETE) rather than the changes to physical pages. Row operations in the transaction log are identified by primary key or unique index value, so that if you recover a database, the updated values for that operation in the transaction log can be applied to the correct row. If no primary key or unique index exists, then all of the values in the row are used as the row's primary key. This behavior can result in significant transaction log growth. A table that does not have an index may result in slow database recovery times.

Related Information

Transaction Log Size Considerations [page 300]
Indexes
The Transaction Log [page 292]

1.6.5.4.4 Performance Warning: Page Size Too Small for Database %1

The performance warning for the page size being too small for database 1% is displayed if two conditions are met.

1. The database page size is 2 KB.
2. The number of pages in the entire database is greater than 100000, corresponding to a database size of approximately 200 MB.

It is recommended that you unload the database and then reload it into a new database with a 4 KB or larger page size for greater space efficiency and better performance.

Related Information

Database Rebuilds

1.6.5.4.5 Performance Warning: Database File %1 Consists of %2 Disk Fragments

When a database starts running on a database server, SQL Anywhere reports the number of file fragments for the main database file, as reported by the operating system.

Excessive disk fragmentation of the disk volume that holds the database can cause performance problems because fragmentation can lead to additional disk latency during I/O operations.

The problem of file system fragmentation is independent of table page fragmentation within the database. The analysis and resolution of these separate problems are performed using different methods and tools. The number of file fragments is reported only for Windows-based SQL Anywhere database servers. Unix file systems, including Linux file systems, incur significantly fewer fragmentation issues than Windows-based systems.

On Windows platforms, database files generally consist of more than one fragment. When you receive this message for a database, the number of fragments relative to the size of the database file helps you determine whether the system administrator needs to take any action. For example, a 100 GB database file that is composed of 25 fragments should not be considered a serious problem, but a 50 MB database file consisting of 300 disk fragments may be affected by performance problems. To eliminate file fragmentation problems,

put the database on a disk partition by itself and then periodically run one of the available Windows disk defragmentation utilities.

1.6.5.4.6 Performance Warning: In Database %3, Table %1 AUTOINCREMENT Column %2 Is Not Indexed

This performance warning is reported when a permanent base table contains an AUTOINCREMENT column and there is no index on the column.

The warning is generated only in cases where the maximum identity value for that column, as stored in the MAX_IDENTITY column in the ISYSTABCOL catalog table, is unknown or if it becomes invalid. This situation can occur if an existing column has been altered to be an AUTOINCREMENT column or if the global autoincrement settings for the database have changed.

Related Information

The AUTOINCREMENT Default
The GLOBAL AUTOINCREMENT Default
SYSTABCOL System View

1.6.5.4.7 Performance Warning: View %1 in Database %2 was Invalidated Due to DDL Operations on One of Its Referenced Objects

This warning is reported when a DDL operation invalidates an existing view.

If a view is dependent on a schema object that has been modified, and the view subsequently fails to recompile, the status of that view remains INVALID. With each subsequent reference from within a query, the database server attempts to recompile the view.

Related Information

Dependencies and Schema-altering Changes
Views
Statuses for Regular Views

1.6.5.5 Troubleshooting Network Communications

There are a few ways of troubleshooting network communications.

Network software involves several different components, increasing the likelihood of problems. Although some tips concerning network troubleshooting are provided here, the primary source of assistance in network troubleshooting should be your IT department or network administrator.

Use Logging

Specifying the -z and -o database server options logs diagnostic communication messages, and other messages to the *Database Server Messages* window for troubleshooting purposes.

On the client, adding the LogFile connection parameter to your connection string will log client-side diagnostic communication messages to a file. These messages can help you diagnose how a connection is failing, what connection parameters are used for the connection attempt, and what communication links are being used.

Adjust Timeout Values

If you are experiencing problems with connections unexpectedly disconnecting, consider adjusting either the liveness or the idle timeout values.

Related Information

-z Database Server Option [page 532]
LogFile (LOG) Connection Parameter [page 97]
LivenessTimeout (LTO) Connection Parameter [page 95]
-tl Database Server Option [page 506]
IdleTimeout (IDLE) Connection Parameter [page 89]
-ti Database Server Option [page 505]

1.7 Database Administration Tools and Utilities

Several tools and methods are provided for administering a database.

In this section:

SQL Central [page 1094]
SQL Central is a graphical tool for managing your database servers, databases, and the objects they contain.

Software Updates [page 1113]
You can check for software updates in several ways.

Database Administration Utilities [page 1115]
Several utility programs for performing database administration tasks are provided.

1.7.1 Interactive SQL

Interactive SQL is a tool that lets you execute SQL statements, run SQL script files, as well as view and compare plans.

Interactive SQL supports the following databases:

- SAP HANA
- SAP IQ
- SAP SQL Anywhere
- UltraLite

In this section:

Starting Interactive SQL [page 1030]
Use Interactive SQL to connect to several different kinds of databases.

Executing SQL Statements (Interactive SQL) [page 1032]
Execute multiple SQL statements from Interactive SQL.

Result Sets (Interactive SQL) [page 1035]
Once you execute a query in Interactive SQL, you can sort and edit the result set.

SQL Statements for Interactive SQL [page 1046]
Several SQL statements are only supported by Interactive SQL.

Creating a Graphical Plan with Detailed and Node Statistics (Interactive SQL) [page 1047]
The Plan Viewer is a graphical tool for viewing graphical plans and text plans for UltraLite databases.

Customizing Interactive SQL [page 1049]
Configure how Interactive SQL result sets are displayed, disable warning messages, and set Interactive SQL as the default editor for `.sql` files.

Query Editor (Interactive SQL) [page 1090]
The Query Editor is a tool in Interactive SQL that helps you build SELECT statements.

Expression Editor (Interactive SQL) [page 1093]
The Expression Editor helps you create search conditions, computed columns, and subqueries.

Administration Tools Security (Interactive SQL) [page 1094]
There are a number of security features available within the administration tools that protect your information.

1.7.1.1 Starting Interactive SQL

Use Interactive SQL to connect to several different kinds of databases.

Prerequisites

Linux

You must have a version of Linux that supports the Linux desktop icons, and they must have been installed when you installed SQL Anywhere.

Unix and macOS

The SQL Anywhere utilities must already be sourced.

Procedure

Open Interactive SQL and connect to a database.

Option	Action
Command line and/or Unix	Run the following command: `dbisql` If you do not include the -c option, which specifies the connection parameters for the database, or if you supply insufficient connection parameters, the *Connect* window appears where you can enter connection information for the database.
Windows	1. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *Interactive SQL*. 2. Enter the connection information for your database in the *Connect* window. 3. Click *Connect*.
SQL Central	1. Click *Tools* > *SQL Anywhere 17* > *Open Interactive SQL*. 2. Enter the connection information for your database in the *Connect* window. 3. Click *Connect*. → Tip You can also use one of the following methods to access Interactive SQL from SQL Central: • Selecting a database, and clicking *File* > *Open Interactive SQL*. • Right-clicking a database, and clicking *Open Interactive SQL*. • Right-clicking a stored procedure, and clicking *Execute From Interactive SQL*. Interactive SQL opens with a CALL to the procedure in the *SQL Statements* pane and executes the stored procedure.

Option	Action
	• Right-clicking a table or view and clicking *View Data In Interactive SQL*. Interactive SQL opens with a `SELECT * FROM table-name` and executes the query.
macOS	1. In the Finder, double-click *Interactive SQL* in `/Applications/SQLAnywhere17`. 2. Enter the connection information for your database in the *Connect* window. 3. Click *Connect*.
Linux	1. Click *Applications* > *SQL Anywhere 17* > *Administration Tools* > *Interactive SQL*. 2. Enter the connection information for your database in the *Connect* window. 3. Click *Connect*.

Results

Interactive SQL is started and you are connected to the database.

Example

- To start Interactive SQL and connect to the sample database from the command line, run the following command:

```
dbisql -c "UID=DBA;PWD=sql;DSN=SQL Anywhere 17 Demo"
```

- To start Interactive SQL and connect to an SAP HANA database, run the following command:

```
dbisql -hana -c "user=username;password=userpasswd;host=hana-server:30015"
```

Related Information

Alphabetical List of Connection Parameters [page 45]
UNIX and Linux Environment Variables [page 567]
Interactive SQL Utility (dbisql) [page 1179]

1.7.1.2 Executing SQL Statements (Interactive SQL)

Execute multiple SQL statements from Interactive SQL.

Procedure

1. In Interactive SQL, type your queries in the *SQL Statements* pane. When you enter more than one statement, each statement must end with a command delimiter, which is a semicolon (;) by default, or the keyword GO on its own line.
2. Choose one of the following options to execute the SQL statements:

Option	Action
Execute all SQL statements	Click *SQL* > *Execute*.
Execute selected SQL statements	Select the statements, and then click *SQL* > *Execute Selection*.
Execute SQL statements one at a time	Place your cursor in the statement that you want to execute, and then click *SQL* > *Single Step*.

3. If an error occurs for a SQL Anywhere or UltraLite database, an error window appears. The error is also logged on the *History* tab for reference.

Results

Interactive SQL shows the result sets from the statements. By default, row numbers appear to the left of the result set.

In this section:

Related Information

Administration Tools Configuration

Returning Multiple Result Sets
Customizing Interactive SQL [page 1049]
command_delimiter Option [Interactive SQL] [page 1059]
Troubleshooting Unexpected Symbols When Viewing Data [page 608]

1.7.1.2.1 Statement Canceling (Interactive SQL)

A cancel operation stops the current processing.

The *Stop* button on the Interactive SQL toolbar cancels a statement.

If a SQL script file was being processed, you are prompted for an action to take (*Stop Command File*, *Continue*, or *Exit Interactive SQL*). These actions can be controlled with the Interactive SQL on_error option.

Related Information

on_error Option [Interactive SQL] [page 1072]

1.7.1.2.2 Executing SQL Script Files (Interactive SQL)

Use Interactive SQL to open, view, and save script files. SQL script files are text files that contain SQL statements and are used to execute SQL statements repeatedly.

Procedure

SQL script files can be executed in any of the following ways:

Option	Action
Use the Interactive SQL READ statement	For example, the following statement executes the file `temp.sql`: `READ temp.sql;`
Load and run the script file	1. Click *File* > *Open*. 2. Select the script file you want to load.
Run the script file without loading it	*File* > *Run Script*.

Option	Action
Supply a script file as a command-line argument for Interactive SQL	For example, the following command runs the SQL script file `myscript.sql` against the sample database: `dbisql -c "DSN=SQL Anywhere 17 Demo;PWD=sql" myscript.sql`

Related Information

SQL Script Files
Managing the Favorites List [page 1052]
READ Statement [Interactive SQL]

1.7.1.2.3 Opening Multiple Interactive SQL Tabs

Open multiple Interactive SQL tabs, each of which corresponds to a separate database connection.

Context

You can connect simultaneously to two (or more) different databases on different database servers, or you can open concurrent connections to a single database.

Procedure

1. In Interactive SQL, click *Window* > *New Tab*.

   → Tip

   If the SQLCONNECT environment variable is set, or if you are already connected to a database, the database server attempts to use this information to connect to a database before it prompts you for information.

   Likewise, if the ULSQLCONNECT environment variable is set, or if you are already connected to an UltraLite database, the database server attempts to use this information to connect to a database before it prompts you for information in. If these attempts fail, or if you are not already connected to a database, the *Connect* tab appears.

2. In the *Connect* tab, enter the connection information for your database, and click *Connect*.

   You can also connect to or disconnect from a database by clicking *SQL* > *Connect* or *SQL* > *Disconnect*.

Results

The connection information (including the database name, your user ID, and the database server name) appears in the Interactive SQL title bar.

Related Information

UltraLite Connection Parameters and the ULSQLCONNECT Environment Variable
SQLCONNECT Environment Variable [page 582]
CONNECT Statement [ESQL] [Interactive SQL]
DISCONNECT Statement [ESQL] [Interactive SQL]

1.7.1.3 Result Sets (Interactive SQL)

Once you execute a query in Interactive SQL, you can sort and edit the result set.

To sort a result set, click a column header in the *Results* tab to sort the results by that column. If the *Results* tab does not contain the entire result set, you are prompted to fetch the remaining results. Otherwise, only the currently retrieved results are sorted.

You can also select rows from the result set and copy them for use in other applications. The field delimiter, quoting character, and escape character for the results are controlled by the isql_field_separator, isql_quote, and isql_escape_character options, respectively. These options can be viewed and changed in *Options* window in Interactive SQL or by executing the SET OPTION statement.

Interactive SQL supports editing, inserting, and deleting rows. Editing the result set has the same effect as executing UPDATE, INSERT, and DELETE statements. After editing a result set, the equivalent INSERT, UPDATE, and DELETE statements are added to the Interactive SQL command history.

To edit a row or value in the result set, you must have the proper privileges on the table or column you want to modify values from. For example, to delete a row, then you must have DELETE privilege for the table the row belongs to.

You cannot edit a result set if you:

- Select columns from a table with a primary key, but do not select all the primary key columns.
- Attempt to edit the result set of a JOIN (for example, if there is data from more than one table in the result set).
- Attempt to edit a table that has its editing disabled.

Editing the result set may fail if you:

- Attempt to edit a row or column you do not have privileges on.
- Enter an invalid value (for example, a string in a numeric column or a NULL in a column that does not allow NULLs).

When editing fails, an Interactive SQL error message appears explaining the error, and the database table values remain unchanged.

In this section:

Related Information

Interactive SQL Options [page 1054]
Viewing Interactive SQL History [page 1051]

1.7.1.3.1 Editing a Row in an Interactive SQL Result Set

Edit values in database tables using Interactive SQL.

Prerequisites

You must have the UPDATE privilege on the columns being modified.

For SQL Anywhere and UltraLite databases, table editing must not be disabled.

Context

When you edit a result set, you can only make changes to the values in one row at a time.

Procedure

1. Execute a simple SELECT query on a table in Interactive SQL.

   The query can only select columns from one table and the query cannot contain expressions.
2. On the *Results* tab, click the value you want to change.
3. Right-click the value and click *Edit Row*, or press F2.

   A ... button appears in the table cell containing the value.
4. Click ... and choose one of the following options:

   Edit In Window

   Opens a window where you can type a new value (character data fields only).

   Set To NULL

   If the column is not nullable, this menu item does not appear.

   Set To DEFAULT

   This menu item appears when adding a new row to a table, and only if the column has defined a default value.

   Load From File

   Lets you browse for a file and then loads the contents of the cell from the file.

   Enter the new value. To change other values in the row, press Tab or Shift+Tab to move to the other values.
5. Press Enter to update the database once you are done editing values in the row.

   Press Esc to cancel the change that was made to the selected value.
6. Execute a COMMIT statement to make your changes to the table permanent.

Results

The row in the result set is edited.

Related Information

Administration Tools Configuration

1.7.1.3.2 Disabling Table Editing in Interactive SQL

Disable table editing via the *Options* window in Interactive SQL.

Context

When deploying Interactive SQL, you can disable table editing.

Procedure

1. In Interactive SQL, click *Tools* > *Options*, and then click either *SQL Anywhere*, *UltraLite*, *IQ*, or *SAP HANA*.
2. Ensure that *Scrollable Table* is selected and click *Disable Editing*.
3. Click *OK*.
4. Execute a query.

   You must execute a new query for the changes to table editing to take effect.

Results

Table editing is disabled.

1.7.1.3.3 Inserting a New ow into an Interactive SQL Result Set

Add rows to a table within Interactive SQL.

Prerequisites

You must have the INSERT object-level privilege on the table to add rows.

You must have executed a query to have a result set.

Context

To add values to a row, tab between columns in the result set.

You cannot enter invalid data types into a column. For example, you cannot enter a string into a column that accepts the INT data type.

Procedure

1. In Interactive SQL, right-click the result set and click *Add Row*.

   A new blank row appears with a blinking cursor in the first value in the row.
2. Enter the new value and then press Tab to move to the next column.

Option	Action
Insert values into columns with default values	When a column accepts a default value, the cell contains *(DEFAULT)*. Click *(DEFAULT)* to insert the default value. Similarly, if a column accepts NULL values, *(NULL)* appears in the cell. If a column cannot be NULL and does not have a default value, you must enter a value.
Insert values into computed columns	If the result set contains a computed column and you do not specify a value for the computed column, the value is calculated when the database is updated. However, if you specify a value for the computed column, the database is updated with the specified value, and a value is not calculated for the computed column.

   Repeat this step until all the column values are added.
3. Press Enter to update the database.
4. Execute a COMMIT statement to make your changes to the database permanent.

Results

You have inserted a new row into the database.

Related Information

INPUT Statement [Interactive SQL]
input_format Option [Interactive SQL] [page 1063]

1.7.1.3.4 Deleting a Row from an Interactive SQL Result Set

Delete a row from a result set in Interactive SQL.

Prerequisites

You must have DELETE privilege on the table to delete rows.

You must have executed a query to have a result set.

Procedure

1. In Interactive SQL, select the row(s) you want to delete. To select a row(s):
   - Press and hold the Shift key while clicking the row(s).
   - Press and hold the Shift key while using the Up or Down Arrow.

   To delete non-consecutive rows, you must delete each row individually. To select multiple disjoint rows in a list, press Control+Click.
2. Press Delete.

   The selected row(s) are removed from the database table.
3. Execute a COMMIT to make the change permanent.

Results

The row is deleted from the result set.

1.7.1.3.5 Copying Rows, Columns, and Cells from an Interactive SQL Result Set

Copy cells, rows, and columns from a result set in Interactive SQL and paste them into other applications.

Context

When copying complete rows, column headings and row values are copied to the clipboard. String values are enclosed in the quote character. If a string value contains the quote character, then it is doubled. Row values are formatted according to the following Interactive SQL options:

- isql_escape_character option [Interactive SQL]
- isql_field_separator option [Interactive SQL]
- isql_quote option [Interactive SQL]

You can also change these options from Interactive SQL by clicking *Options* > *Import/Export*. If these options are set to their defaults, the copied row data is comma-delimited and string values are enclosed in single quotes.

When copying columns, column headings are included but string values are not enclosed in the quote character.

When copying cells, column headings are not included and string values are not enclosed in the quote character.

Procedure

In Interactive SQL, retrieve the result set you want to copy.

Option	Action
Copy one row	Right-click any cell in the row and click *Copy Data* > *Rows*.
Copy multiple rows	Hold the Ctrl key while clicking cells in the rows, and then right-click and click *Copy Data* > *Rows*. The selected row(s), including their column headings, are copied to the clipboard.
Copy one column	Right-click any cell in the column and click *Copy Data* > *Columns*.
Copy multiple columns	Hold the Ctrl key while clicking cells in the columns, and then right-click and click *Copy Data* > *Columns*. If the *Results* pane does not contain the entire result set, you are prompted to fetch the remaining results before selecting them. Otherwise, only those results that have been fetched so far are selected.

Option	Action
Copy a cell	Right-click the cell you want to copy and click *Copy Data* > *Cells*.
Copy multiple cells	Hold the Ctrl key while clicking cells, and then right-click and click *Copy Data* > *Cells*. When you do this, no column headings are copied. Only the data is copied to the clipboard, and no quoting is done.

Results

The selected data is copied.

Next Steps

You can paste the selected information into other applications.

Related Information

isql_escape_character Option [Interactive SQL] [page 1066]
isql_field_separator Option [Interactive SQL] [page 1068]
isql_quote Option [Interactive SQL] [page 1071]

1.7.1.3.6 Generating a SQL Statement from Interactive SQL Result Sets

Create INSERT, DELETE, and UPDATE statements for selected rows in a result set.

Prerequisites

You must have executed a query to have a result set to perform the task on.

Procedure

1. Select the row(s) you want to generate a statement for.
2. Right-click the selection, click *Generate*, and then click *INSERT Statement*, *DELETE Statement*, or *UPDATE Statement*.

Results

The statement is copied to the clipboard.

Next Steps

Execute the statement generated. You must have the necessary privileges to execute the INSERT, DELETE, or UPDATE statement.

1.7.1.3.7 Viewing Images (Interactive SQL)

Use Interactive SQL to view images, including SVGs, BLOBs, and images that are less than 5000000 pixels.

Prerequisites

You must have the privileges necessary to execute your query.

Your image must be in one of the following formats: BMP (Windows 95 version), GIF, JPG, PNG, SVG, or WBMP (wireless bitmap).

Procedure

1. From Interactive SQL, connect to the database.
2. Execute a query that returns a result set that contains an image.
3. In the result set, click in a cell that contains *(IMAGE)*, click ..., and then click *View in Window*.

   When viewing SVGs, use the following shortcuts:

Action	Option
Ctrl+Left Mouse Button Drag	Drags a boundary so that you can adjust the point of view.

Action	Option
Ctrl+Right Mouse Button Drag	Rotates the image around its center.
Shift+Left Mouse Button Drag	Pans the image to a new location.
Shift+Right Mouse Button Drag	Zooms the image in and out.

The *Value of Column* `column-name` window appears.

4. Click *OK* to close the window.

Results

The image is viewed.

Example

Connect to the sample database and execute the following statement to return JPEG images:

```
SELECT * FROM GROUPO.Products;
```

The following window appears when you click the JPEG image associated with the cotton cap:

Connect to the sample database and execute the following statement to return an SVG image:

```
SELECT '<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="100%" height="100%" version="1.1"
xmlns="http://www.w3.org/2000/svg">
<rect width="99" height="99"
style="fill:rgb(0,0,255);stroke-width:1;
stroke:rgb(0,0,0)"/>
</svg>';
```

Related Information

Viewing Spatial Data as Images (Interactive SQL)
Viewing Spatial Data as Images (Spatial Viewer)

1.7.1.3.8 Viewing HTML and XML Data (Interactive SQL)

View HTML and XML data from the result set of a query in a separate window in Interactive SQL.

Prerequisites

You must have the privileges necessary to execute your query.

Procedure

1. In Interactive SQL, connect to the database.
2. Execute a query that returns a result set that contains HTML or XML data.
3. In the results, select a cell that contains HTML or XML content, click ..., and then click *View in Window*.

Results

The *Value of Column* `column-name` window appears.

Example

To view a sample result set that contains HTML, run the following query:

```
SELECT '<html>
 <head>
  <meta http-equiv=Content-Type content="text/html; charset=windows-1252">
   <title>Tee Shirt</title>
 </head>
 <body lang=EN-US>
  <p>
   <span  style=font-size:10.0pt;font-family:Arial>
   We have improved the design of this perennial favorite.
   A sleek and technical shirt built for the trail, track, or
   sidewalk. UPF rating of 50+.
   </span>
  </p>
```

```
 </body>
</html>';
```

To view a sample result set that contains a well-formed XML document, run the following query:

```
SELECT '<?xml version="1.0" encoding="UTF-8"?>
<parent>
    <children>
        <child name="Tim" />
        <child name="Katie" />
    </children>
</parent>';
```

Next Steps

Click *OK* to close the window.

1.7.1.4 SQL Statements for Interactive SQL

Several SQL statements are only supported by Interactive SQL.

- CLEAR statement
- CONFIGURE statement
- CONNECT statement
- DESCRIBE statement
- DISCONNECT statement
- EXIT statement
- HELP statement
- INPUT statement
- OUTPUT statement
- PARAMETERS statement
- READ statement
- SET CONNECTION statement
- SET OPTION statement
- START SERVER statement
- START LOGGING statement
- STOP LOGGING statement
- SYSTEM statement

Related Information

CLEAR Statement [Interactive SQL]

CONFIGURE Statement [Interactive SQL]
CONNECT Statement [ESQL] [Interactive SQL]
DESCRIBE Statement [Interactive SQL]
DISCONNECT Statement [ESQL] [Interactive SQL]
EXIT Statement [Interactive SQL]
HELP Statement [Interactive SQL]
INPUT Statement [Interactive SQL]
OUTPUT Statement [Interactive SQL]
PARAMETERS Statement [Interactive SQL]
READ Statement [Interactive SQL]
SET CONNECTION Statement [Interactive SQL] [ESQL]
SET OPTION Statement [Interactive SQL]
START SERVER Statement [Interactive SQL]
START LOGGING Statement [Interactive SQL]
STOP LOGGING Statement [Interactive SQL]
SYSTEM Statement [Interactive SQL]

1.7.1.5 Creating a Graphical Plan with Detailed and Node Statistics (Interactive SQL)

The Plan Viewer is a graphical tool for viewing graphical plans and text plans for UltraLite databases.

Procedure

1. Open Interactive SQL and connect to the database.
2. Click *Tools* > *Plan Viewer* or press Shift+F5.

   The Plan Viewer appears in a separate window. The Plan Viewer window is divided into the following panes:

   SQL pane

   This pane provides a place for you to type SQL statements that you want to generate plans for.

   Results pane

   This pane shows the graphical plan, and is only for SQL Anywhere databases.

   Details pane

   This pane provides text details about the plan for SQL Anywhere databases.

   For UltraLite databases, this pane shows the text plan.
3. Type a statement in the *SQL* pane.
4. In the *Statistics level* list, click *Detailed and node statistics*.
5. Click *Get Plan* to generate a plan for the specified query.
6. Click *Save As*.
7. Specify where you want to save the plan and type a file name. Click *Save*.

Results

A graphical plan with *Detailed and node statistics* is created.

Next Steps

Open the graphical plan by clicking *Tools* > *Plan Viewer*. Click *Open*. Select a plan file (`.saplan`), and then click *Open*.

In this section:

Customizing a Graphical Plan [page 1048]
Customize the appearance of items in the graphical plan.

Related Information

Graphical Plans
Viewing an Execution Plan
Viewing a Graphical Plan

1.7.1.5.1 Customizing a Graphical Plan

Customize the appearance of items in the graphical plan.

Procedure

1. Open the Plan Viewer. In Interactive SQL, click *Tools* > *Plan Viewer*.
2. Open a plan. Right-click the plan in the lower left pane of the Plan Viewer and click *Customize*.
3. Change the settings.
4. Click *OK* when finished.
5. Click *Get Plan*.

Results

The customized graphical plan is generated.

1.7.1.6 Customizing Interactive SQL

Configure how Interactive SQL result sets are displayed, disable warning messages, and set Interactive SQL as the default editor for `.sql` files.

Context

Configure settings for the tabs and panes in Interactive SQL by using the *Options* window. When you deploy Interactive SQL, you can control which features are shown or enabled in the *Options* window.

Procedure

Open Interactive SQL and connect to a database.

Option	Action
Customize how result sets are displayed	1. Choose to show the results as a scrollable table or as text. Click *Data* and then choose one of the following: **Show Results As Scrollable Table** You can edit the result set in this format. This is the default. **Show Results As Text** This option displays the result set as text using a monospace font. The result set is not editable in this format. 2. Execute a query. You must execute a new query for the changes to table editing to take effect. 3. Right-click within the result set and choose one of the following: **Size Columns To Data** The table columns widen so that the values are not truncated. **Size Columns To Window** The table columns are made the same width so they all fit within the window without a horizontal scroll bar.
Disable warning messages	You can disable some of the warning messages that appear in Interactive SQL. For example, you can suppress the warning that appears when you have unsaved text in the *SQL Statements* pane and you click *File* > *Exit*. 1. Click *Tools* > *Options* > *Messages*. 2. Clear the messages listed in the *Optional Messages* list.
Set Interactive SQL as the default editor for `.sql` files	1. Click *Tools* > *Options* > *General*. 2. Click *Make Interactive SQL The Default Editor For .SQL Files And Plan Files*.

Option	Action
	3. Click *OK*.
Configure the *Execute Statements* toolbar button to execute all or selected statements	1. Click *Tools* > *Options* > *Toolbar*. 2. Select one of the following options: • To execute all SQL statements, click *Execute*. This is the default setting. • To execute only the selected SQL statements, click *Execute Selection*.

In this section:

Viewing Interactive SQL History [page 1051]
View the SQL Statement history that Interactive SQL automatically saves when you execute a statement.

Looking up Tables, Columns, and Procedures (Interactive SQL) [page 1053]
Look up tables, columns, and procedures in Interactive SQL.

Interactive SQL Options [page 1054]
Use the SET OPTION statement to change the values of several Interactive SQL options.

Keyboard Shortcuts (Interactive SQL) [page 1077]
Interactive SQL provides several keyboard shortcuts.

Text Completion (Interactive SQL and SQL Central) [page 1082]
Interactive SQL and SQL Central provide a text completion option that can supply you with complete keywords and object names as you type.

Logging Statements (Interactive SQL) [page 1084]
Log statements in Interactive SQL.

Automatically Release Schema Locks (Interactive SQL) [page 1085]
Configure Interactive SQL to automatically release database schema locks if there are no uncommitted changes.

Source Control Integration (Interactive SQL) [page 1086]
Interactive SQL integrates with third-party source control systems, allowing you to perform many common source control operations on files from within Interactive SQL.

Related Information

Administration Tools Configuration
SQL Script Files
SQL Script Files
Troubleshooting Unexpected Symbols When Viewing Data [page 608]

1.7.1.6.1 Viewing Interactive SQL History

View the SQL Statement history that Interactive SQL automatically saves when you execute a statement.

Context

When you execute a SQL statement that contains password information such as CREATE USER, CONNECT, or CREATE EXTERNLOGIN, then the password is obfuscated in the *History* at all times.

When the statement history is viewed in subsequent Interactive SQL sessions, passwords are replaced with "..."

Procedure

1. Execute a SQL statement.
2. On the *History* tab, use the *Show* dropdown menu to choose whether to show statements from *SQL run in this tab*, *SQL run on demo*, or *SQL run on all databases*.
3. On the *History* tab, select the statement(s) and then choose one of the following options:

Option	Action
Insert SQL	Right-click the statement, and then click *Send to* *SQL Editor*.
Save your history	Right-click the statement, then click "Save History" in the context menu.
Add a statement to your *Favorites* list	Right-click the statement(s), and then click *Send to* *Favorite*.
Delete a statement	• Right-click the statement(s), and then click *Delete* *Selected Statements*. • To delete all history, right-click the statement(s), and then click *Delete* *History*.

In this section:

Managing the Favorites List [page 1052]
Create, edit, or share your list of favorites.

Related Information

Options Window: History Tab

1.7.1.6.1.1 Managing the Favorites List

Create, edit, or share your list of favorites.

Context

A favorites list is specific to a single user, and cannot be seen by other users unless you share it with them. When you export a favorites list, it is saved as a `.fav` file.

Procedure

Choose one of the following options:

Option	Action
Add a file, statement, or connection to your favorites list	1. Open the SQL script file that you want to add to your favorites. 2. Click *Favorites* > *Add to Favorites*. 3. Click *Add the open file '`filename`'*. 4. Specify a name for the `.sql` file in the *Name* field.
Add a SQL statement to your favorites	1. Click *Favorites* > *Add to Favorites*. 2. Click *Add SQL Statements*, and then specify a name for the favorite in the *Name* field.
Add a connection to your favorites	1. Click *Favorites* > *Add to Favorites*. 2. Click *Save The Connection Password*, and then specify a name for the favorite in the *Name* field.
Edit the name of the file	1. Click *Favorites* > *Show Favorites*. This opens a *Favorites* pane that appears on the left side of the Interactive SQL window. 2. Right-click a favorite, and then click *Edit*. 3. Edit the statement, and then click *Save*.
Share your favorites list	Click *Favorites*, and then choose one of the following options: • To export your favorites list: 1. Click *Export Favorites*. 2. Specify a name for the `.fav` file, and then click *Export*. • To import a favorites list: 1. Click *Import Favorites*. 2. Browse to the favorites list that you want to import, and then click *Import*.

Related Information

Administration Tools Configuration

1.7.1.6.2 Looking up Tables, Columns, and Procedures (Interactive SQL)

Look up tables, columns, and procedures in Interactive SQL.

Procedure

1. In Interactive SQL, click *Tools*, and then choose one of the following:

Option	Action
Look up table names	Click *Lookup Table Name* or press F7. Find and select the table.
Look up column names	Click *Lookup Table Name* or press F7. Find and select the table containing the column. Click *Show Columns*.
Look up procedure names	Click *Lookup Procedure Name* or press F8. Find and select the procedure.

→ Tip

- Enter the first few characters of a table or procedure name to narrow the list of items.
- Use the SQL wildcard characters % (percent sign) to match any string of zero or more characters, and '_' (underscore) to match any one character.
- Search for a wildcard character within a table name by prefixing it with an escape character. The escape character for the SQL Anywhere ODBC driver and SQL Anywhere JDBC driver is ~ (tilde).
- When you are looking up the name of a procedure, but are unsure if it is the procedure you want, click the "More" button to see the source of the procedure.

2. Click *OK* to insert the name into the *SQL Statements* pane at the current cursor position.

Results

The table, column, or procedure name is found and inserted into the *SQL Statements* pane.

Related Information

Text Completion (Interactive SQL and SQL Central) [page 1082]

1.7.1.6.3 Interactive SQL Options

Use the SET OPTION statement to change the values of several Interactive SQL options.

Option	Values	Default
auto_commit option [Interactive SQL]	On, Off	Off
auto_refetch option [Interactive SQL]	On, Off	On
bell option [Interactive SQL]	On, Off	On
command_delimiter option [Interactive SQL]	String	' ; '
commit_on_exit option [Interactive SQL]	On, Off	On
default_isql_encoding option [Interactive SQL]	String	Empty string
echo option [Interactive SQL]	On, Off	On
input_format option [Interactive SQL]	TEXT, EXCEL, FIXED	TEXT
isql_allow_read_client_file option [Interactive SQL]	On, Off, Prompt	Prompt
isql_allow_write_client_file option [Interactive SQL]	On, Off, Prompt	Prompt
isql_command_timing option [Interactive SQL]	On, Off	Off
isql_escape_character option [Interactive SQL]	Character	' \ '
isql_field_separator option [Interactive SQL]	String	' , '
isql_maximum_displayed_rows option [Interactive SQL]	All or a non-negative integer	500
isql_print_result_set option [Interactive SQL]	Last, All, None	Last
isql_quote option [Interactive SQL]	String	'
nulls option [Interactive SQL]	String	'(NULL)'
on_error option [Interactive SQL]	Stop, Continue, Prompt, Exit, Notify_Continue, Notify_Stop, Notify_Exit	Prompt
output_format option [Interactive SQL]	TEXT, EXCEL, FIXED, HTML, SQL, XML	TEXT
output_length option [Interactive SQL]	Integer	0

Option	Values	Default
output_nulls option [Interactive SQL]	String	Empty string
truncation_length option [Interactive SQL]	Integer	256

In this section:

nulls Option [Interactive SQL] [page 1072]
Specifies how NULL values in the database appear when displaying results in Interactive SQL.

on_error Option [Interactive SQL] [page 1072]
Controls what happens if an error is encountered while executing statements in Interactive SQL.

output_format Option [Interactive SQL] [page 1073]
Sets the default output format for data retrieved by a SELECT statement that is redirected to a file or output using the OUTPUT statement.

output_length Option [Interactive SQL] [page 1075]
Controls the length of column values when Interactive SQL exports information to an external file.

output_nulls Option [Interactive SQL] [page 1075]
Controls the way NULL values are exported.

truncation_length Option [Interactive SQL] [page 1076]
Controls the truncation of wide columns for displays to fit on a screen.

Related Information

SET OPTION Statement [Interactive SQL]

1.7.1.6.3.1 auto_commit Option [Interactive SQL]

Controls whether a COMMIT is performed after each statement.

Allowed Values

On, Off

Default

Off

Remarks

If auto_commit is On, a database COMMIT is performed after each successful statement.

By default, a COMMIT or ROLLBACK is performed only when the user issues a COMMIT or ROLLBACK statement, except when a SQL statement performs an automatic commit (such as the CREATE TABLE statement).

> **i** Note
>
> Setting the Interactive SQL auto_commit option does not set the database server auto_commit option.

Example

To disable automatic commit by Interactive SQL, execute the following statement.

```
set option auto_commit='Off';
```

The current setting of the Interactive SQL auto_commit option can be viewed from the *Execution* tab of the *SQL Anywhere* options.

Related Information

commit_on_exit Option [Interactive SQL] [page 1060]
SET OPTION Statement [Interactive SQL]
COMMIT Statement
ROLLBACK Statement
Keyboard Shortcuts (Interactive SQL) [page 1077]

1.7.1.6.3.2 auto_refetch Option [Interactive SQL]

Controls whether query results are fetched again after deletes, updates, or inserts.

Allowed Values

On, Off

Default

On

Remarks

If auto_refetch is On, the current query results that appear on the *Results* tab in the Interactive SQL *Results* pane are refetched from the database after any INSERT, UPDATE, or DELETE statement. Depending on how complicated the query is, this may take some time. For this reason, it can be turned off.

Related Information

SET OPTION Statement [Interactive SQL]

1.7.1.6.3.3 bell Option [Interactive SQL]

Controls whether the bell sounds when an error occurs.

Allowed Values

On, Off

Default

On

Remarks

Set this option according to your preference.

Related Information

SET OPTION Statement [Interactive SQL]

1.7.1.6.3.4 command_delimiter Option [Interactive SQL]

Sets the string that indicates the end of a statement in Interactive SQL.

Allowed Values

String

Default

Semicolon (;)

Remarks

In general, there is no need to change the command delimiter. You should leave it as a semicolon.

An alternative to using a semicolon or another string as a statement delimiter is to enter the separator *go* on a line by itself, at the beginning of the line.

Specifying *go* on its own line at the beginning of the line is always recognized as a command delimiter, even if you set the command_delimiter option to a different value.

The command_delimiter value can be any string of characters with the following restrictions:

- If the delimiter contains any one of & (ampersand), * (asterisk), @ (at sign), : (colon), . (period), = (equals), ((left parentheses),) (right parentheses), or | (vertical bar), the delimiter must not contain any other character. For example, * is a valid delimiter, but ** is not.
- You should not use an existing keyword as a command separator.
- The command delimiter can be any sequence of characters (including numbers, letters, and punctuation), but it cannot contain embedded blanks.
  If the command delimiter is set to a string beginning with a character that is valid in identifiers, the command delimiter must be preceded by a space. The command delimiter is case sensitive. You must enclose the new command delimiter in single quotation marks in a SET OPTION statement. When the command delimiter is a semicolon (the default), no space is required before the semicolon.

Example

The following example sets the command delimiter to a tilde:

```
SET OPTION command_delimiter='~';
MESSAGE 'hello'~
```

You can also use the Interactive SQL -d option to set the command delimiter without including a SET OPTION command_delimiter statement in a `.sql` file. For example, if you have a script file named `test.sql` that uses tildes (~) as the command delimiter, you could run:

```
dbisql -d "~" test.sql
```

Related Information

Batches
Keywords
Interactive SQL Utility (dbisql) [page 1179]
SET OPTION Statement [Interactive SQL]

1.7.1.6.3.5 commit_on_exit Option [Interactive SQL]

Controls the behavior when Interactive SQL disconnects or shuts down.

Allowed Values

On, Off

Default

On

Remarks

Controls whether a COMMIT or ROLLBACK is done when you leave Interactive SQL. When commit_on_exit is set to On, a COMMIT is done.

Related Information

SET OPTION Statement [Interactive SQL]
ROLLBACK Statement
COMMIT Statement

Keyboard Shortcuts (Interactive SQL) [page 1077]

1.7.1.6.3.6 default_isql_encoding Option [Interactive SQL]

Specifies the encoding that should be used by READ, INPUT, and OUTPUT statements.

Allowed Values

Identifier or string

Default

Use system encoding (empty string)

Scope

Can be set as a temporary option only, for the duration of the current connection.

Remarks

This option is used to specify the encoding to use when reading or writing files. It cannot be set permanently. The default encoding is the code page for the platform you are running on. On English Windows computers, the default code page is 1252.

When running Interactive SQL, the encoding that is used by the INPUT, OUTPUT, or READ statement is determined in the following order:

- The encoding specified by the ENCODING clause of the INPUT, OUTPUT, or READ statement (if this clause is specified).
- The encoding specified with the default_isql_encoding option (if this option is set).
- The default encoding for the platform you are running on. On English Windows computers, the default encoding is 1252.

If the input file was created using the OUTPUT statement and an encoding was specified, then the same ENCODING clause should be specified on the INPUT statement.

Example

Set the encoding to UTF-16 (for reading Unicode files):

```
SET TEMPORARY OPTION default_isql_encoding = 'UTF-16';
```

Set the encoding to cp437 I (Windows English OEM):

```
SET TEMPORARY OPTION default_isql_encoding = 'cp437';
```

Related Information

Character Sets [page 601]
International Languages and Character Sets [page 595]
OUTPUT Statement [Interactive SQL]
READ Statement [Interactive SQL]
INPUT Statement [Interactive SQL]
SET OPTION Statement [Interactive SQL]

1.7.1.6.3.7 echo Option [Interactive SQL]

Permits fine control over logging of SQL statements and messages to a file when logging is enabled using the START LOGGING statement.

Allowed Values

On, Off

Default

On

Remarks

This option is most useful when you use the READ statement to execute a SQL script file or when you run a script file in Interactive SQL by clicking *File* > *Run Script*. Logging must be enabled for this option to take effect.

Related Information

START LOGGING Statement [Interactive SQL]
SET OPTION Statement [Interactive SQL]

1.7.1.6.3.8 input_format Option [Interactive SQL]

Sets the default data format expected by the INPUT statement.

Allowed Values

TEXT

Input lines are assumed to be text characters, one row per line, with values separated by commas or semicolons. Alphabetic strings can be enclosed in apostrophes (single quotes) or quotation marks (double quotes).

The default delimiter for locales that use a period as a decimal point is a comma (,). For locales that use a comma as a decimal point, the default delimiter is a semicolon (;). Field delimiters containing commas or semicolons must be enclosed in either single or double quotes. If single or double quotes are used, then double the quote character to use it within the string.

EXCEL

The first row on the sheet is assumed to contain the column names.

FIXED

Input lines are in fixed length format.

Default

TEXT

Related Information

INPUT Statement [Interactive SQL]
SET OPTION Statement [Interactive SQL]

1.7.1.6.3.9 isql_allow_read_client_file Option [Interactive SQL]

Controls whether client file reads are permitted for the connection.

Allowed Values

On, Off, Prompt

Default

Prompt

Remarks

This option controls whether the database server can read files on the client computer. On means reading is allowed. Off means reading is not allowed. Prompt means prompt the user for the action to take.

This option is stored on a per-connection basis and persists only for the duration of the connection. You can set this option using the SET TEMPORARY OPTION statement. If you omit the TEMPORARY keyword, Interactive SQL reports an error.

This option allows a data file to be read without user intervention when a LOAD TABLE is executed from a stored procedure or trigger.

You must have the READ CLIENT FILE system privilege to read a file on a client computer.

Related Information

Access to Data on Client Computers
Client-side Data Security
READ_CLIENT_FILE Function [String]
SET OPTION Statement [Interactive SQL]
LOAD TABLE Statement
allow_read_client_file Option [page 674]
allow_write_client_file Option [page 676]
isql_allow_write_client_file Option [Interactive SQL] [page 1065]

1.7.1.6.3.10 isql_allow_write_client_file Option [Interactive SQL]

Controls whether client file writes are permitted for the connection.

Allowed Values

On, Off, Prompt

Default

Prompt

Remarks

This option controls whether the database server can write to files on the client computer. On means writing is allowed. Off means writing is not allowed. Prompt means prompt the user for the action to take.

This option is stored on a per-connection basis and persists only for the duration of the connection. You can set this option using the SET TEMPORARY OPTION statement. If you omit the TEMPORARY keyword, Interactive SQL reports an error.

You must have the WRITE CLIENT FILE system privilege to write to a file on a client computer.

Related Information

Access to Data on Client Computers
Client-side Data Security
WRITE_CLIENT_FILE Function [String]
SET OPTION Statement [Interactive SQL]
UNLOAD Statement
allow_write_client_file Option [page 676]
allow_read_client_file Option [page 674]
isql_allow_read_client_file Option [Interactive SQL] [page 1064]

1.7.1.6.3.11 isql_command_timing Option [Interactive SQL]

Controls whether the execution time is printed when Interactive SQL is run as a console application in interactive mode.

Allowed Values

On, Off

Default

On

Remarks

This option has no effect when Interactive SQL is run as a windowed application or as a console application in non-interactive (batch) mode.

Related Information

START LOGGING Statement [Interactive SQL]
SET OPTION Statement [Interactive SQL]

1.7.1.6.3.12 isql_escape_character Option [Interactive SQL]

Controls the escape character used in place of unprintable characters in data exported to text files.

Allowed Values

Any single character

Default

A backslash (\ )

Remarks

When Interactive SQL exports strings that contain unprintable characters (such as a carriage return), it converts each unprintable character into a hexadecimal format and precedes it with an escape character. The character you specify for this setting is used in the output if your OUTPUT statement does not contain an ESCAPE CHARACTER clause. This setting is used only if you are exporting to a text file.

Example

Create a table that contains one string value with an embedded carriage return (denoted by the "\n" in the INSERT statement). Then export the data to `c:\escape.txt` with a # sign as the escape character.

```
CREATE TABLE escape_test( text varchar(10 ) );
INSERT INTO escape_test VALUES( 'one\ntwo' );
SET OPTION isql_escape_character='#';
SELECT * FROM escape_test;
OUTPUT TO c:\escape.txt FORMAT TEXT;
```

This code places the following data in `escape.txt`:

```
'one#x0atwo'
```

The number sign (#) is the escape character and x0a is the hexadecimal equivalent of the \n character.

The start and end characters (in this case, single quotation marks) depend on the isql_quote setting.

Related Information

isql_quote Option [Interactive SQL] [page 1071]
SET OPTION Statement [Interactive SQL]

1.7.1.6.3.13 isql_field_separator Option [Interactive SQL]

Controls the default string used for separating (or delimiting) values in data exported to files.

Allowed Values

String

Default

For locales that use a period as a decimal separator, the default field separator is a comma (,). For locales that use a comma as a decimal separator, the default field separator is a semicolon (;).

Remarks

If an OUTPUT statement does not contain a DELIMITED BY clause, the value of this setting is used.

Example

- The first example sets the field separator to a colon in the data exported to `c:\Employees.txt`.

  ```
  SET OPTION isql_field_separator=':';
  SELECT Surname, GivenName FROM Employees WHERE EmployeeID < 150;
  OUTPUT TO c:\Employees.txt FORMAT TEXT;
  ```

  This code places the following data in `Employees.txt`:

  ```
  'Whitney': 'Fran'
  'Cobb':'Matthew'
  'Chin':'Philip'
  'Jordan':'Julie'
  ```

  The start and end characters (in this case, single quotation marks) depend on the isql_quote setting.
- The next example sets the field separator to a tab in the data exported to `c:\Employees.txt`.

  ```
  SET OPTION isql_field_separator='\t';
  SELECT Surname, GivenName FROM Employees WHERE EmployeeID < 150;
  OUTPUT TO c:\Employees.txt FORMAT TEXT;
  ```

  This code places the following data in `Employees.txt`:

  ```
  'Whitney' 'Fran'
  'Cobb' 'Matthew'
  ```

```
'Chin' 'Philip'
'Jordan' 'Julie'
```

The start and end characters (in this case, single quotation marks) depend on the isql_quote setting. The escape character (in this case the backslash) depends on the isql_escape_character setting.

Related Information

isql_quote Option [Interactive SQL] [page 1071]
isql_escape_character Option [Interactive SQL] [page 1066]
SET OPTION Statement [Interactive SQL]

1.7.1.6.3.14 isql_maximum_displayed_rows Option [Interactive SQL]

Specifies the maximum number of rows that can appear in the *Results* pane in Interactive SQL.

Allowed Values

ALL or a non-negative integer

Default

500

Remarks

This option lets you specify the maximum number of rows that appear in the *Results* pane.

You can also set the value for this option in the *Options* window in Interactive SQL (*Maximum number of rows to display*).

If the isql_maximum_displayed_rows option is set to a value less than the number of rows in the result set, then at most isql_maximum_displayed_rows are fetched and displayed in the *Results* pane. If an OUTPUT statement is used to log the result set to a file, then the query is re-executed so that the entire result set can be logged. If re-execution is undesirable, then set isql_maximum_displayed_rows to ALL.

Similarly, if the truncation_length option for columns that appear in the *Results* pane is set to a value less than the actual column data width, then the column output is truncated. If an OUTPUT statement is used to log the

result set to a file, then the query is re-executed so that the entire result set can be logged. If re-execution is undesirable, then set truncation_length to a sufficiently large value.

> ⚠ Caution
>
> Interactive SQL can run out of memory when displaying large result sets. If this problem occurs, Interactive SQL reports the problem but will not display the result set.

Related Information

SET OPTION Statement [Interactive SQL]

1.7.1.6.3.15 isql_print_result_set Option [Interactive SQL]

Specifies which result set(s) are printed when a `.sql` file is run.

Allowed Values

LAST

Prints the result set from the last statement in the file.

ALL

Prints result sets from each statement in the file which returns a result set.

NONE

Does not print any result sets.

Default

LAST

Remarks

The isql_print_result_set option takes effect only when you run Interactive SQL as a command line program (for example, when running a `.sql` file).

This option allows you to specify which result set(s) are printed when a `.sql` file is run.

1.7.1.6.3.16 isql_quote Option [Interactive SQL]

Controls the default string that begins and ends all strings in data exported to text files.

Allowed Values

String

Default

A single apostrophe (')

Remarks

Controls the default string that begins and ends all strings in data exported to text files. If an OUTPUT statement does not contain a QUOTE clause, this value is used by default.

Example

To change the default string that begins and ends all strings to a double quote character.

```
SET OPTION isql_quote='"';
SELECT Surname, GivenName FROM Employees WHERE EmployeeID < 150;
OUTPUT TO c:\Employees.txt FORMAT TEXT;
```

This code places the following data in `Employees.txt`:

```
"Whitney", "Fran"
"Cobb","Matthew"
"Chin","Philip"
"Jordan","Julie"
```

The separator characters (in this case, commas) depend on the isql_field_separator setting.

Related Information

isql_field_separator Option [Interactive SQL] [page 1068]
SET OPTION Statement [Interactive SQL]

1.7.1.6.3.17 nulls Option [Interactive SQL]

Specifies how NULL values in the database appear when displaying results in Interactive SQL.

Allowed Values

String

Default

(NULL)

Remarks

Set this option according to your preference. This value is not used when saving result sets to a file. The value used when saving to a file is specified by the output_nulls option.

Related Information

SET OPTION Statement [Interactive SQL]
output_nulls Option [Interactive SQL] [page 1075]

1.7.1.6.3.18 on_error Option [Interactive SQL]

Controls what happens if an error is encountered while executing statements in Interactive SQL.

Allowed Values

Stop

Interactive SQL stops executing statements.

Prompt

Interactive SQL prompts the user to see if they want to continue.

Continue

The errors are sent to the *History* tab in the *Results* area Interactive SQL, and Interactive SQL continues executing statements.

Exit

Interactive SQL shuts down.

Notify_Continue

The error is reported, and the user is prompted to press Enter or click *OK* to continue.

Notify_Stop

The error is reported and the user is prompted to press Enter or click *OK* to stop executing statements.

Notify_Exit

The error is reported and the user is prompted to press Enter or click *OK* to shut down Interactive SQL.

Default

Prompt

Remarks

When you are executing a `.sql` file, the values Stop and Exit are equivalent. If you specify either of these values, Interactive SQL shuts down.

This option is ignored if the Interactive SQL utility (dbisql) onerror option is specified.

Related Information

Interactive SQL Utility (dbisql) [page 1179]
SET OPTION Statement [Interactive SQL]

1.7.1.6.3.19 output_format Option [Interactive SQL]

Sets the default output format for data retrieved by a SELECT statement that is redirected to a file or output using the OUTPUT statement.

Allowed Values

TEXT

The output is a TEXT format file with one row per line in the file. By default, values are separated by the field separator, and strings are enclosed in apostrophes (single quotes). The field separator is controlled by the *isql_field_separator_option*.

EXCEL

The output is a Microsoft Excel file, where the column headings are written to the first row on the worksheet.

FIXED

The output is fixed format, with each column having a fixed width. No column headings are output in this format.

HTML

The output is in HTML format.

SQL

The output is the Interactive SQL INPUT statement that is required to recreate the information in the table.

XML

The output is an XML file encoded in UTF-8 and containing an embedded DTD. Binary values are encoded in CDATA blocks with the binary data rendered as 2-hex-digit strings.

Remarks

The output_format option is used when the OUTPUT statement does not specify the FORMAT clause. The output_format option is ignored if the OUTPUT statement includes a FORMAT clause.

Default

TEXT

Related Information

SET OPTION Statement [Interactive SQL]
OUTPUT Statement [Interactive SQL]

1.7.1.6.3.20 output_length Option [Interactive SQL]

Controls the length of column values when Interactive SQL exports information to an external file.

Allowed Values

Non-negative integer

Default

0 (no truncation)

Remarks

This option controls the maximum length of column values when Interactive SQL exports data to an external file (using output redirection with the OUTPUT statement). This option affects only the TEXT, HTML, and SQL output formats.

Related Information

SET OPTION Statement [Interactive SQL]

1.7.1.6.3.21 output_nulls Option [Interactive SQL]

Controls the way NULL values are exported.

Allowed Values

String

Default

Empty string

Remarks

This option controls the way NULL values are written by the OUTPUT statement. Every time a NULL value is found in the result set, the string from this option is returned instead. This option affects only the TEXT, HTML, FIXED, and SQL output formats.

Example

The following example changes the value that appears for NULL values to *(unknown)*:

```
SET OPTION output_nulls = '(unknown)';
```

Related Information

SET OPTION Statement [Interactive SQL]
IFNULL Function [Miscellaneous]

1.7.1.6.3.22 truncation_length Option [Interactive SQL]

Controls the truncation of wide columns for displays to fit on a screen.

Allowed Values

Integer

Default

256

Remarks

The truncation_length option limits the length of displayed column values. The unit is in characters. A value of 0 means that column values are not truncated. The default truncation length is 256.

You can also set the value for this option in the *Options* window in Interactive SQL (*Truncation length*).

If the truncation_length option for columns that appear in the *Results* pane is set to a value less than the actual column data width, then the column output is truncated. If an OUTPUT statement is used to log the result set to a file, then the query is re-executed so that the entire result set can be logged. If re-execution is undesirable, then set truncation_length to a sufficiently large value.

Similarly, If the isql_maximum_displayed_rows option is set to a value less than the number of rows in the result set, then at most isql_maximum_displayed_rows are fetched and displayed in the *Results* pane. If an OUTPUT statement is used to log the result set to a file, then the query is re-executed so that the entire result set can be logged. If re-execution is undesirable, then set isql_maximum_displayed_rows to ALL.

Related Information

SET OPTION Statement [Interactive SQL]

1.7.1.6.4 Keyboard Shortcuts (Interactive SQL)

Interactive SQL provides several keyboard shortcuts.

Key(s)	Description
Alt+F4	Shuts down Interactive SQL.
Alt+Left Arrow On macOS, use Control+Command+Left Arrow.	Displays the previous SQL statement in the history list.
Alt+Right Arrow On macOS, use Control+Command+Right Arrow.	Displays the next SQL statement in the history list.
Ctrl+Backspace	Deletes the word to the left of the cursor.
Ctrl+Break	Interrupts the SQL statement that is being executed. This shortcut is only supported on Windows operating systems.
Ctrl+A	Selects all text in the active pane. In the *Results* pane, if the results are not the entire result set, you are prompted to fetch the remaining rows. Otherwise the currently fetched results are selected.

Key(s)	Description
Ctrl+C	In the *Results* pane, copies the selected row(s) and column headings to the clipboard. In the *SQL Statements* pane, copies the selected text to the clipboard.
Ctrl+Shift+C Does not apply on macOS	Executes a Commit statement. **i Note** Executing a COMMIT via the *SQL* menu or a keyboard shortcut does not modify the contents of the *SQL Statements* pane; however, the *Results* tab in the *Results* pane is cleared.
Ctrl+Del	Deletes the word to the right of the cursor.
Ctrl+End	Moves the cursor to the bottom of the current pane.
Ctrl+F	Opens the *Find/Replace* window.
Ctrl+Shift+F10	Displays a popup menu for a cell that has focus in a result set in the *Results* tab. This keyboard shortcut is an alternative to clicking ... within a table cell.
Ctrl+G	Moves the cursor to the specified line in the *SQL Statements* pane.
Ctrl+H On macOS, use Control+Command+H.	Displays the history of your executed SQL statement(s).
Ctrl+Home	Moves the cursor to the top of the current pane.
Ctrl+K	Opens the **Locking Viewer** window.
Ctrl+L	Deletes the current line from the *SQL Statements* pane and puts the line onto the clipboard.
Ctrl+Shift+L	Deletes the current line.
Ctrl+N	Clears the contents of the Interactive SQL window and closes the current file (if any).
Ctrl+O	Opens a file.
Ctrl+P	Prints the contents of the *SQL Statements* pane. Alternatively, click *Tools* > *Options* > *Editor* > *Print*.
Ctrl+Q On macOS use, use Command+Q	Opens the Query Editor. The Query Editor helps you build SQL queries. When you have finished building your query, click *OK* to export it back into the *SQL Statements* pane. On macOS, this shortcut closes Interactive SQL.

Key(s)	Description
Ctrl+Shift+R Does not apply on macOS.	Executes a Rollback statement. **Note** Executing a ROLLBACK via the *SQL* menu or a keyboard shortcut does not modify the contents of the *SQL Statements* pane; however, the *Results* tab in the *Results* pane is cleared.
Ctrl+S	Saves the contents of the *SQL Statements* pane to the specified file.
Ctrl+U	Changes the selection to lowercase characters.
Ctrl+T	Creates a new tab.
Ctrl+Shift+T	Creates a new tab without opening the *Connect* window.
Ctrl+Shift+U	Changes the selection to uppercase characters.
Ctrl+V	Pastes the selected text.
Ctrl+X	Cuts the selected text.
Ctrl+Y	Repeats the last operation.
Ctrl+Z	Undoes the last operation.
Ctrl+]	Moves the cursor to the matching brace. Use this shortcut to match parentheses, braces, brackets, and angle brackets.
Ctrl+Shift+]	Extends the selection to the matching brace. Use this shortcut to match parentheses, braces, brackets, and angle brackets.
Ctrl+Minus Sign (-)	Adds and removes the double-hyphen (--) SQL comment indicator. To turn existing text into comments, select the text in the *SQL Statements* pane and press Ctrl+minus sign. The SQL comment indicator is added to the beginning of each line in the selection. If no text is selected, the comment indicator is added to the beginning of the current line. To remove a comment indicator, select the text and press Ctrl+minus sign.

Key(s)	Description
Ctrl+Forward Slash (/)	Adds and removes the double-slash (//) SQL comment indicator. To turn existing text into comments, select the text in the *SQL Statements* pane and press Ctrl+forward slash. The SQL comment indicator is added to the beginning of each line in the selection. If no text is selected, the comment indicator is added to the beginning of the current line. To remove a comment indicator, select the text and press Ctrl+forward slash.
Ctrl+Up Arrow	Selects the SQL statement preceding the statement that contains the cursor in the *SQL Statements* pane.
Ctrl+Down Arrow	Selects the SQL statement following the statement that contains the cursor in the *SQL Statements* pane.
Ctrl+Period (.)	Selects the entire SQL statement containing the cursor in the *SQL Statements* pane. Alternatively, double-click a line to select the entire SQL statement.
Ctrl+Comma (,)	Selects the line containing the cursor in the *SQL Statements* pane.
Ctrl+Shift+Period (.)	Increases the line indentation of selected text in the *SQL Statements* pane. If no text is selected, the indentation is applied to the current line.
Ctrl+Shift+Comma (,)	Decreases line indentation of selected text in the *SQL Statements* pane. If no text is selected, the indentation is applied to the current line.
Esc	By default, pressing the Esc key has no affect. However, you can set the Esc key to clear the *SQL Statements* pane and close any opened result sets. Click *Tools* > *Options* > *Compatibility* and click *ESCAPE key Clears SQL Statements And Closes Result Sets*.
F1	Opens online help.
F2	Edits the selected value in the result set. You can tab from column to column within the row.
F3	Finds the next occurrence of the specified text.
Shift+F3	Finds the previous occurrence of the selected text.

Key(s)	Description
F5 On macOS, Command+R	Executes all text in the *SQL Statements* pane. You can also perform this operation by clicking *Execute All SQL Statements* on the toolbar or by clicking *SQL* ▸ *Execute*.
Shift+F5	Opens the Plan Viewer for the specified statement in the *SQL Statements* pane. The specified statement is not executed; to execute the statement in the Plan Viewer, click *Get Plan*.
F6	Changes the focus from the *SQL Statements* pane to the *Results* pane and vice versa. When the *Favorites* pane is opened, the focus changes between the *SQL Statements*, *Results*, and *Favorites* panes.
F7	Displays the *Lookup Table Name* window. In this window, you can find and select a table and then press Enter to insert the table name into the *SQL Statements* pane at the cursor position. Or, with a table selected in the list, press F7 again to display the columns in that table. You can then select a column and press Enter to insert the column name into the *SQL Statements* pane at the cursor position.
F8	Displays the *Lookup Procedure Name* window. In this window, you can find and select a procedure and then press Enter to insert the procedure name into the *SQL Statements* pane at the cursor position.
F9 On macOS, Command+Option+R	Executes the text that is selected in the *SQL Statements* pane. If no text is selected, all the statements are executed. You can also perform this operation by clicking *Execute Selected Statements* on the toolbar or by clicking *SQL* ▸ *Execute*.
Shift+F9 On macOS, Control+R	Executes the selected SQL statement, and then selects the next statement. This shortcut allows you to step through a series of SQL statements.
Shift+F10	Displays the shortcut menu for the area that has focus. This keyboard shortcut is an alternative to right-clicking an area.
F11	Opens the *Connect* window if Interactive SQL is not connected to a database.
F12	Disconnects Interactive SQL from the current database.

Key(s)	Description
Home	Moves the cursor to the start of the current line or to the first word on the current line.
Shift+Home	Extends the selection to the start of the text on the current line.
Page Down	Moves a page down in the current pane.
Page Up	Moves a page up in the current pane.

Related Information

SQL Statements for Interactive SQL [page 1046]
Copying Rows, Columns, and Cells from an Interactive SQL Result Set [page 1041]
Text Completion Keyboard Shortcuts [page 1083]
Comments

1.7.1.6.5 Text Completion (Interactive SQL and SQL Central)

Interactive SQL and SQL Central provide a text completion option that can supply you with complete keywords and object names as you type.

For example, typing the letter S causes a text completion window to appear that includes a list of possible keywords or object names beginning with the letter S. You can configure the text completion settings from the *Options* window in Interactive SQL or a text editor window in SQL Central.

Example

In Interactive SQL, type the first letter of a database object name in the *SQL Statements* pane.

By default, the text completion window opens automatically as you type. Alternatively, you can press Ctrl +Space or Ctrl+Shift+Space to open the window. The text completion window contains a list of database objects that begin with the letter(s) you typed.

If you do not see the object name you want, then press Ctrl+A to view a complete list of database objects (based on the filtering options you set). By default, all database objects appear in the list.

Select the object name from the list and then press Enter.

The object name appears in the *SQL Statements* pane.

In this section:

Text Completion Keyboard Shortcuts [page 1083]

Several keyboard shortcuts are available when the text completion list is open.

1.7.1.6.5.1 Text Completion Keyboard Shortcuts

Several keyboard shortcuts are available when the text completion list is open.

Key	Description
Ctrl+A	Shows a context-free list of matches.
Ctrl+Double quote (") (or Ctrl+Shift+')	Completes the name, enclosing it in quotation marks, regardless of the setting of the quoted_identifier option.
Ctrl+Asterisk (*) (or Ctrl+Shift+8)	For tables, inserts a comma-separated list of columns including data types. For stored procedures, inserts the procedure name, followed by a comma-separated list of parameter names and their data types.
Ctrl+C	Changes the contents of the text completion list to show or hide columns.
Ctrl+F	Changes the contents of the text completion list to show or hide SQL functions.
Ctrl+P	Changes the contents of the text completion list to show or hide stored procedures and functions.
Ctrl+Plus Sign (+) (or Ctrl+Shift+=)	For tables, inserts a comma-separated list of columns. For stored procedures, inserts the procedure name, followed by a comma-separated list of parameter names.
Ctrl+S	Changes the contents of the text completion list to show or hide system objects.
Ctrl+Shift+Space or Ctrl+Space	Opens the text completion window. You can also use Ctrl +Space to open the text completion window.
Ctrl+T	Changes the contents of the text completion list to show or hide tables.
Ctrl+V	Changes the contents of the text completion list to show or hide views.
Esc	Closes the text completion window without adding any text.
Tab	Accepts the selection and closes the text completion window.

Related Information

quoted_identifier Option [page 804]

1.7.1.6.6 Logging Statements (Interactive SQL)

Log statements in Interactive SQL.

Context

Statements are recorded until you stop the logging process or end the current session.

You can start or stop logging using either the *SQL* menu, or SQL statements.

The recorded statements are stored in a SQL statements log file so you can use the statements again.

Once you start logging, all SQL statements that you try to execute are logged, including ones that do not execute properly.

Procedure

1. Open Interactive SQL.
2. To start logging:

Option	Action
SQL **menu**	1. Click *SQL* > *Start Logging*. 2. In the *Save As* window, specify a location and name for the file you are logging to. For example, name the file `mylogs.sql`. 3. Click *Save* when finished.
START LOGGING statement	Execute a START LOGGING statement. For example, to start logging to a file named `c:\mylogs.sql`, execute the following statement: `START LOGGING 'c:\mylogs.sql';`

3. To stop logging:

Option	Action
SQL **menu**	Click *SQL* > *Stop Logging*.
STOP LOGGING statement	Execute the following statement: `STOP LOGGING;`

Results

Statement logging in Interactive SQL is started or stopped.

Related Information

Logging SQL Statements [page 338]
START LOGGING Statement [Interactive SQL]
STOP LOGGING Statement [Interactive SQL]
isql_command_timing Option [Interactive SQL] [page 1066]

1.7.1.6.7 Automatically Release Schema Locks (Interactive SQL)

Configure Interactive SQL to automatically release database schema locks if there are no uncommitted changes.

Context

The database server creates schema locks on tables that you view in Interactive SQL, even if you do not modify the table.

In Interactive SQL, you can view the locks for the current connection by clicking *Tools* > *Locking Viewer*.

Procedure

1. In Interactive SQL, click *Tools* > *Options* > *SQL Anywhere*.
2. Select *Scrollable table*, and then click *Automatically release database locks*.

Results

When a statement is executed, Interactive SQL checks if your connection has any uncommitted changes in the database. If none exist, then Interactive SQL releases your schema locks.

Related Information

IdleTimeout (IDLE) Connection Parameter [page 89]
blocking_others_timeout Option [page 695]

1.7.1.6.8 Source Control Integration (Interactive SQL)

Interactive SQL integrates with third-party source control systems, allowing you to perform many common source control operations on files from within Interactive SQL.

On Windows, Interactive SQL integrates with most source control products that support the Microsoft Common Source Code Control API (SCC), including Microsoft Visual SourceSafe. To use source control products that do not support the SCC API on Windows and other operating systems, specify a command line to run for each of the source control actions. Output from those commands appears in a log window.

Interactive SQL supports the following tasks (as long as the task is supported in the source control product):

- Open a source control project
- Get
- Check in
- Check out
- Undo check out
- Compare versions
- Show file history
- Show file properties
- Run the source control manager

If the underlying source control program does not support an action, its corresponding menu item is disabled. For example, Microsoft Visual SourceSafe supports all of these actions, but using a custom (command line) source control system does not support opening a source control project, or running a source control manager.

You should be familiar with the operations of your source control program before attempting to use it from Interactive SQL.

In this section:

Configuring Interactive SQL to Use Source Control [page 1087]
Configure Interactive SQL to use source control before performing source control actions on files, such as checking files in and out, comparing different versions of a file, or viewing the history for a file.

Source Control Projects and Interactive SQL [page 1088]
Some source control products require you to open a source control project before you can perform any other source control actions.

Checking Out Files Using Interactive SQL [page 1088]
Check a file out by modifying its contents in the *SQL Statements* pane or by using the command on the *File* menu.

Additional Source Control Actions [page 1089]

In addition to opening source control projects, and checking files in and out, Interactive SQL supports several other source control actions.

1.7.1.6.8.1 Configuring Interactive SQL to Use Source Control

Configure Interactive SQL to use source control before performing source control actions on files, such as checking files in and out, comparing different versions of a file, or viewing the history for a file.

Prerequisites

Your source control system must be set up.

If you are running Interactive SQL on a Windows computer that has a source control product that supports the Microsoft SCC API, you can use that product or use a custom (command line oriented) system.

Procedure

1. In Interactive SQL, click *Tools* > *Options*.
2. In the left pane, click *Source Control*.
3. Click *Enable source control integration*.

Option	Action
Configure SCC source control systems	Click *OK*.
Configure other source control systems	1. Click *Configure*. 2. In the *Custom Source Control Options* window, click *Reset*. 3. Select your source control system from the list, and then click *OK*. 4. Edit the commands in the list as necessary by selecting an action from the *Source control actions* list, and then typing the corresponding command in the *Command line* pane. When you are defining commands for your system in the *Source control actions* list, use the placeholder [FILENAME] to represent the name of the file that is used when you run the command. For example, the command to submit a file in Perforce is `p4 submit [FILENAME]`. Only the actions that appear bold in this list have commands defined for them. If you do not specify a command line for an action, the item in the *File* > *Source Control* menu is disabled. → Tip You can export your source control command lines to an external file by clicking *Export* in the *Custom Source Control Options* window (accessed by clicking *Tools* > *Options*,

Option	Action
	and then clicking *Configure* on the *Source Control* pane). You can later read these commands back in by clicking *Import* in this window. This may be useful if you need to configure Interactive SQL source control command lines on several computers. 5. Click *OK*, and then click *OK* again.

Results

The source control system is integrated.

1.7.1.6.8.2 Source Control Projects and Interactive SQL

Some source control products require you to open a source control project before you can perform any other source control actions.

The exact definition of what a project is depends on the source control system you are using. Typically, it is a set of files that are under source control, along with a location on your local file system where working copies of the files are placed. You usually have to provide some credentials, such as a user ID and password, to the source control system to open a project.

If your source control system supports opening a source control project, the *File* > *Source Control* > *Open Source Control Project* menu item is enabled. Choosing this option from the *File* menu opens a source control-specific window for opening a project. Once you open a project, you do not have to open it again, even in subsequent Interactive SQL sessions. The project is opened automatically for you.

1.7.1.6.8.3 Checking Out Files Using Interactive SQL

Check a file out by modifying its contents in the *SQL Statements* pane or by using the command on the *File* menu.

Context

When you configure the source control options for Interactive SQL, choose *Automatically Check Out Files When Editor Contents Are Modified* to have Interactive SQL attempt to check out a file when you modify its contents in the *SQL Statements* pane.

Procedure

1. In Interactive SQL, click *File* > *Open*, and then browse to the file you want to open.

   > ⚠ Caution
   >
   > If you are using a Source Control Code API-compliant source control system, the status is always accurate. If you use the custom source control system, the status is based on whether the file is read-only or not. A read-only file is assumed to be checked in, but editable files could be either checked out or not controlled.

   The file status appears on the status bar at the bottom of the Interactive SQL window. The status is one of *Checked In*, *Checked Out*, or *Not Controlled*. Files that are checked in are assumed to be read-only, and *Read-Only* appears in the Interactive SQL title bar. The file in the following example is checked in:
2. Check out the file by clicking *File* > *Source Control* > *Check Out*.

Results

Depending on which source control product you are using, you may be prompted for a comment or other options as part of the check out procedure.

Next Steps

When you are finished making edits to your file, you can check it back in from Interactive SQL. Click *File* > *Source Control* > *Check In*. Enter check-in comments if you are prompted.

1.7.1.6.8.4 Additional Source Control Actions

In addition to opening source control projects, and checking files in and out, Interactive SQL supports several other source control actions.

The availability of these actions depends on the source control system you are using. Click *File* > *Source Control*, and then choose an action:

Get

This action gets the latest copy of the file you currently have open in the *SQL Statements* pane.

Undo Check Out

If you have checked out a file, and you want to discard your changes, click *File* > *Source Control* > *Undo Checkout*. This discards your working copy of the file, and then downloads the copy of the file that is in the source control archive.

Compare Versions

This action compares the working copy of the file you have opened against the version in the source control archive.

History

This action displays a list of source control actions (typically check-ins) that have been made to the file you have open.

Properties

This action displays a list of source control properties that are associated with the file you have opened.

Run Source Control Manager

This action launches the management program for your source control system. For example, if you are using Microsoft Visual SourceSafe, this launches Microsoft Visual SourceSafe Explorer.

1.7.1.7 Query Editor (Interactive SQL)

The Query Editor is a tool in Interactive SQL that helps you build SELECT statements.

You can create SQL queries in the Query Editor, or you can import queries and edit them. When you have finished your query, click *OK* to export it back into SQL Central or Interactive SQL for processing.

Using SQL Statements with the Query Editor

You do not need to use SQL statements to create queries with the Query Editor. However, you can use SQL statements with the Query Editor in the following ways:

- You can create a query in the *SQL Statements* pane in Interactive SQL, and import it into the Query Editor by highlighting the code before you open the editor.
- At any time while using the Query Editor, you can click *SQL* at the bottom of the window to see the SQL statements for the query you are building. You can directly edit the code, and the fields are automatically updated in the Query Editor.

Configuring the Query Editor

You can configure the Query Editor from Interactive SQL or SQL Central so that the SQL is fully formed, meaning that all table and column names are fully qualified and names are quoted. This extra formatting is not normally necessary, but it ensures that the SQL works in all situations. You can also choose to get a list of tables on startup.

To configure the Query Editor from Interactive SQL click *Tools* > *Options* > *SQL Anywhere*, and then select the *Query Editor* tab.

Query Editor Tabs

The Query Editor provides a series of tabs that guide you through the components of a SQL query, most of which are optional. The tabs are presented in the order that SQL queries are usually built:

Tab	Description
Tables tab	Use this tab to specify the tables in your query.
Joins tab	Use this tab to specify a join strategy for combining the data in the tables if you include more than one table in your query. If you do not specify a join strategy, then the Query Editor suggests one. If there is a foreign key relationship between the tables, it generates a join condition based on that relationship, or it suggests a cross product. When you open queries, the Query Editor accepts exactly the join strategy that you specified (and an unspecified JOIN is not defaulted to KEY JOIN, as it would be otherwise in SQL Anywhere).
Columns tab	Use this tab to specify the columns in your result set. If you do not specify columns, all columns appear.
INTO tab	Use this tab to assign results to variables.
WHERE tab	Use this tab to specify conditions for restricting the rows in your result set.
GROUP BY tab	Use this tab to group rows in the result set.
HAVING tab	Use this tab to restrict the rows in your result set based on group values.
ORDER BY tab	Use this tab to sort the rows.

Query Editor Tools

The Query Editor also contains the following tools:

Window	Description
Expression Editor	Use the Expression Editor to build search conditions or define computed columns.
Derived Table	Use this window, which is nearly identical to the main Query Editor, to create derived tables and subqueries.

Each component of the Query Editor has context-sensitive online help that describes how to use the tabs, and provides links to the SQL Anywhere documentation that explains relevant concepts and usage.

Query Editor Limitations

The Query Editor builds SELECT statements. It is not designed to create views, although you can create them in Interactive SQL and reference them in the Query Editor. Nor was it designed to create UPDATE statements or

other non-SELECT SQL statements. It creates a single SELECT statement, so it does not build unions or intersects of SELECT statements. In addition, the Query Editor does not support Transact-SQL syntax.

In this section:

Related Information

Expression Editor (Interactive SQL) [page 1093]
Queries
SELECT Statement

1.7.1.7.1 Creating a Query Using the Query Editor

Create SQL queries in the Query Editor, or import queries and edit them.

Procedure

1. Connect to a database from Interactive SQL.
2. Open the Query Editor.

   Click *Tools* > *Edit Query*.

   If you have a SQL statement selected in Interactive SQL, the selected statement is automatically imported into the Query Editor.
3. Create your query.

   At any time while using the Query Editor, you can click *SQL* at the bottom of the window to see the SQL statement for the query you are building. You can directly edit the code, and the fields are automatically updated in the Query Editor user interface.
4. Click *OK* to write the query to the *SQL Statements* pane.

Results

A query is created using the Query Editor, and is exported to Interactive SQL for processing.

1.7.1.8 Expression Editor (Interactive SQL)

The Expression Editor helps you create search conditions, computed columns, and subqueries.

To edit existing expressions, highlight the expression before you open the Expression Editor. Otherwise, what you create in the Expression Editor is appended to previously existing expressions when you click *OK*.

Components

Expression

This box is where you build your expression.

Columns

This box lists the columns that are in your query. To insert a column into your expression, double-click it here, or type it directly into the *Expression* pane.

Functions

Functions are predefined expressions used to return information about the database. To insert a function into your expression, choose it from the dropdown list.

Stored Procedures

This box lists the stored procedures that you can use.

Numeric keypad

The numeric keypad is located in the bottom left section of the window. Click a number to insert it into the expression. You can also type numbers from your keyboard.

Comparison operators

These arithmetic symbols, such as =, appear in the middle of the bottom section of the window. Click a symbol to insert it into the expression. Only a subset of common operators are included here, but you can type others from your keyboard.

Logical operators

Logical operators, such as AND, appear in the right bottom section of the window. Click an operator to insert it into the expression. This group is a subset of commonly used logical operators. You can use any logical operator by typing it from your keyboard.

Edit A Subquery And Insert The Results Into The Editor button

This button is located with the logical operators. Click it to access a window that guides you in creating subqueries.

Adding a Subquery

To create a subquery, open the Expression Editor and click the *Edit A Subquery And Insert The Results Into The Editor* button (located next to the *NOT* button). This opens the *Subquery* window, which looks identical to the main Query Editor except that the word *Subquery* is in the title bar. You create a subquery the same way that you create a main query.

Related Information

SQL Functions
Logical Operators
Query Editor (Interactive SQL) [page 1090]
Comparison Operators
Search Conditions
Subqueries in Search Conditions

1.7.1.9 Administration Tools Security (Interactive SQL)

There are a number of security features available within the administration tools that protect your information.

Interactive SQL

The -nohistory command line option The -nohistory command-line option prevents Interactive SQL from saving your SQL statement history when the program is terminated. However, SQL statements still appear on the *History* tab while the program is running.

1.7.2 SQL Central

SQL Central is a graphical tool for managing your database servers, databases, and the objects they contain.

SQL Central Key Features

Easy command access

The *File* menu in SQL Central automatically updates when you select an object, providing commands related directly to that object. You can also right-click an object to access these commands.

Task wizards

To add a new object, SQL Central provides you with wizards that walk you through the task step by step.

Drag-and-drop functionality

SQL Central supports drag-and-drop functionality for many operations. For example, to copy tables to a different database, you can click and drag them to that location.

When you copy a database object, the SQL for the object is copied to the clipboard so it can be pasted into other applications, such as Interactive SQL or a text editor. For example, if you copy an index in SQL Central and paste it into a text editor, the CREATE INDEX statement for that index appears.

Keyboard shortcuts

Many commonly used commands have keyboard shortcuts; these shortcuts are listed beside the command names in the menus.

Plug-in support

You can manage a variety of database products and tools by using plug-ins. In SQL Central, click *Help*, and then choose a plug-in name to get additional information about using and configuring the plug-in.

Refreshing the Content

SQL Central caches information about the database. If you make changes to the database on another connection, SQL Central won't see the changes until you refresh the database in SQL Central. For example, products managed from SQL Central may support concurrent access. Another user could change the structure of the folders list while you are examining it in SQL Central.

To ensure that you are always viewing the most up to date information, SQL Central provides the following refresh commands in the *View* menu:

Refresh Folder

Refreshes your view of the currently expanded container. You can also press F5 to refresh the folder.

Refresh All

Refreshes your view of all expanded containers.

These commands query SQL Central and its plug-ins to get up-to-date information.

Deploying SQL Central

Subject to your license agreement, you can deploy SQL Anywhere administration tools, including SQL Central.

In this section:

Related Information

Administration Tool Deployment
MobiLink Plug-in for SQL Central
Creating an UltraLite Database with the Create Database Wizard
SQL Central Keyboard Shortcuts [page 1098]

1.7.2.1 SQL Central Navigation

The SQL Central window is split into two vertically aligned panes.

Left Pane

Choose from the *View* menu what you want the left pane to display:

Folders pane

This pane displays a hierarchical view of database objects.

Folders shows only the containers in the object tree; it does not show objects that are not containers of other objects. For example, the left pane may show a Columns folder (a container), but not the columns themselves because they are items, and appear in the right pane instead.

Configure which database objects are displayed in the *Folders* pane by right-clicking the database, clicking *Configure Type Filter*, and selecting the object types that you want to have displayed in the left pane. Then click *OK*. Optionally, you can click *Make This The Default Type Filter For New Databases* to set the default type filter to the one configured. The default type filter is used whenever you connect to a database that does not have a specified type filter.

Tasks pane

This pane displays a task list for the currently selected database object.

Search pane

This pane allows you to search for objects in a plug-in.

Right Pane

The right pane shows the contents of the currently selected container. The right pane has tabs that display the contents of the container that is selected in the left pane, and other information about the selected container.

Quickly find objects or restrict the contents that appear in this pane by using the *Filter* field at the top. The *Filter* field searches for items that contain the given text.

Configure the columns that appear on a tab in the right pane by clicking *View* > *Choose Columns*.

Change the appearance of the right pane by clicking *Tools* > *Options*.

Toolbar

The main window toolbar provides you with buttons for common commands. To show or hide the toolbar, click *View* > *Toolbars* > *Standard Toolbars*. With the main toolbar, you can:

- navigate through the object folders
- connect to or disconnect from a database, database server, or product plug-in
- show the *Tasks*, *Folders*, or *Search* pane
- access the *Connection Profiles* window (also accessible from the *Tools* menu)
- refresh the view of the current folder
- cut, copy, paste, and delete objects
- undo or redo actions
- view the properties window for a selected object

Context Dropdown List

The *Context* dropdown list, which appears below the toolbar, lets you navigate the object folders for a plug-in.

Status bar

The status bar, which appears at the bottom of the main window, shows a brief summary of menu commands as you navigate through the menus. To show or hide the status bar, click *View* > *Status Bar*.

Related Information

Searching for Database Objects [page 1097]

1.7.2.2 Searching for Database Objects

Search a database within SQL Central for a specified database object, or search for a string within the SQL of a database object.

Prerequisites

If you do not have the necessary privileges to view the *Login Mappings* folder, *Maintenance Plans* folder, or the *Page Usage* or *Locks* tab for a database, any search you perform skips these folders and/or tabs.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to a database.
2. Click *View* > *Search Pane*.

   The *Search* pane appears in the left pane.
3. Configure options for the search.

Option	Action
SQL Anywhere 17 **plug-in**	**Search In SQL** Select this option to include the SQL of procedures, events, functions and triggers in the search. **Search Dynamic Properties (Connections, Statistics, Locks)** Select this option to include dynamic properties, such as connected users, SQL Remote statistics, table locks, and table page usage information in the search.
MobiLink 17 **plug-in**	**Search In Scripts** Select this option to include synchronization scripts in the search.

4. Click *Search*.

Results

The search results appear in *Results* in the left pane. Double-click a result to open it in the right pane.

In this section:

SQL Central Keyboard Shortcuts [page 1098]
SQL Central provides several keyboard shortcuts.

1.7.2.2.1 SQL Central Keyboard Shortcuts

SQL Central provides several keyboard shortcuts.

Function key	Description
Alt+Enter	Opens the properties window for the selected item.
Ctrl+C	Copies the selection to the clipboard.
Ctrl+F	Sets focus to the filter.
Ctrl+V	Inserts the clipboard contents.
Ctrl+X	Cuts the selection and moves it to the clipboard.

Function key	Description
Delete	Deletes the selection.
F1	Opens the SQL Central help.
F5	Refreshes the contents of the selected folder.
F9	Opens the *Connection Profiles* window.
F11	Opens the *Connection* menu if there are multiple plug-ins loaded. If only one plug-in is loaded, pressing F11 opens the *Connect* window for that plug-in.
F12	Disconnects when there is only one connection in SQL Central. When there is more than one connection, pressing F12 opens the *Disconnect* window where you can select the connection you want to disconnect.
Shift+F10	Opens the popup menu for the selected object.

1.7.2.3 Code Editor

The Code Editor appears as a *SQL* tab in the right pane of SQL Central, as a separate window in SQL Central, and as the *SQL Statements* pane in Interactive SQL.

To open the Code Editor in a separate window in SQL Central, in the left pane, select a database object, and then click *File* > *Edit In New Window*.

Beyond standard text editing functions, the Code Editor provides the following functionality:

- a toolbar and status bar
- automatic syntax highlighting
- language-sensitive indenting
- the ability to find and replace text
- the ability to open from and save to files (the availability of this functionality depends on the plug-in you are using)
- the ability to print the code
- text completion when typing code

In this section:

Code Editor Keyboard Shortcuts [page 1100]
SQL Central provides several keyboard shortcuts for the Code Editor.

1.7.2.3.1 Code Editor Keyboard Shortcuts

SQL Central provides several keyboard shortcuts for the Code Editor.

Function key	Description
Alt+F4	Closes the Code Editor (if a separate window) or closes SQL Central if you are editing text in the right pane of SQL Central.
Backspace	Deletes the selection. If nothing is selected, pressing Backspace deletes the character to the left of the cursor.
Ctrl+]	Moves the cursor to the matching brace. Use this shortcut to match parentheses, braces, brackets, and angle brackets.
Ctrl+A	Selects the entire contents of the Code Editor window.
Ctrl+Backspace	Deletes the word to the left of the cursor.
Ctrl+C	Copies the selected text to the clipboard.
Ctrl+Delete	Deletes the word to the right of the cursor.
Ctrl+End	Moves the cursor to the bottom of the Code Editor window.
Ctrl+F	Opens the *Find/Replace* window where you can search for and replace the specified text if you have not searched for text in the current window. Otherwise, this searches for the next occurrence of the specified text.
Ctrl+F3	Finds the next occurrence of the currently selected text.
Ctrl+G	Opens the *Go To* window where you can specify the line location you want to go to within the Code Editor window.
Ctrl+Home	Moves the cursor to the top of the Code Editor window.
Ctrl+L	Deletes the current line.
Ctrl+Left Arrow	Moves the cursor back one word.
Ctrl+N	Clears the contents of the Code Editor window and closes the current file (if any). This shortcut cannot be used from the *SQL* tab in the right pane of SQL Central.
Ctrl+O	Opens a file when the Code Editor is open as a separate window. This shortcut cannot be used from the *SQL* tab in the right pane of SQL Central.
Ctrl+P	Prints the contents of the Code Editor window. You can configure the appearance of the printed text: click *Tools* > *Options* > *Print*.
Ctrl+Right Arrow	Moves the cursor forward one word.
Ctrl+S	Saves the contents of the Code Editor window.
Ctrl+Shift+]	Extends the selection to the matching brace. Use this shortcut to match parentheses, braces, brackets, and angle brackets.
Ctrl+Shift+End	Extends the selection to the end of the code.

Function key	Description
Ctrl+Shift+F3	Find the previous occurrence of the currently selected text.
Ctrl+Shift+Home	Extends the selection to the beginning of the code.
Ctrl+Shift+L	Deletes the current line.
Ctrl+Shift+Left Arrow	Extends the selection back one word.
Ctrl+Shift+Right Arrow	Extends the selection forward one word.
Ctrl+Shift+U	Changes the selection to uppercase characters.
Ctrl+Shift+Period (.)	Increases the line indentation of selected text in the Code Editor window. If no text is selected, the indentation is applied to the current line.
Ctrl+Shift+Comma (,)	Decreases the line indentation of selected text in the Code Editor window. If no text is selected, the indentation is applied to the current line.
Ctrl+U	Changes the selection to lowercase characters.
Ctrl+V	Inserts the Clipboard contents at the current cursor location.
Ctrl+X	Cuts the selected text.
Ctrl+Y	Redoes the most recently undone action.
Ctrl+Z	Undoes the last action.
Ctrl+Minus Sign (-)	Adds and removes the double-hyphen (--) SQL comment indicator. To turn existing text into comments, select the text in the Code Editor window and press Ctrl+minus sign. The double-hyphen, SQL comment indicator is added to the start of the lines that contain the selected text. If no text is selected, the comment indicator is added to the start of the current line. To remove a comment indicator, select the text and press Ctrl+minus sign.
Ctrl+Forward Slash (/)	Adds and removes the double-slash (//) SQL comment indicator. To turn existing text into comments, select the text in the Code Editor window and press Ctrl+forward slash. The double-slash, SQL comment indicator is added to the start of the lines that contain the selected text. If no text is selected, the comment indicator is added to the start of the current line. To remove a comment indicator, select the text and press Ctrl+forward slash.

Function key	Description
Delete	Deletes the selection.
Down Arrow	Moves the cursor down one line.
End	Moves the cursor to the end of the current line.
F3	Opens the *Find/Replace* window where you can search for and replace the specified text if you have not searched for text in the current window. Otherwise, this searches for the next occurrence of the specified text.
Home	Move the cursor to the start of the current line or to the start of the text on the current line.
Left Arrow	Moves the cursor one character to the left.
Page Down	Moves the cursor to the end of the current page.
Page Up	Moves the cursor to the top of the current page.
Right Arrow	Moves the cursor one character to the right.
Shift+Down Arrow	Extends the selection down one line.
Shift+End	Selects the current line.
Shift+F3	Opens the *Find/Replace* window where you can search for and replace the specified text if no text is selected. If text is selected, finds the previous occurrence of the selected text.
Shift+F10	Opens the popup menu for the area that has focus. This keyboard shortcut is an alternative to right-clicking an area.
Shift+Home	Extends the selection to the start of the text on the current line.
Shift+Left Arrow	Extends the selection one character to the left of the currently selected character(s).
Shift+Page Down	Extends the selection down one page.
Shift+Page Up	Extends the selection up one page.
Shift+Right Arrow	Extends the selection one character to the right of the currently selected character(s).
Shift+Up Arrow	Extends the cursor up one line.
Up Arrow	Moves the cursor up one line.

Related Information

Comments

1.7.2.4 SQL Central Plug-in Architecture

Each SAP product is managed by a separate plug-in in SQL Central.

The following SQL Central plug-ins are provided:

- SQL Anywhere databases
- MobiLink synchronization
- UltraLite databases

The plug-ins for these products must be registered and loaded before you can use the products in SQL Central. In most cases, when you install the product its plug-in is automatically registered and loaded. For special situations, you can also manually register the plug-ins that appear in SQL Central.

> → Tip
>
> SQL Central has its own version number, separate from the plug-in version numbers. To see the SQL Central version number or a plug-in's version number, click *Help* > *About SQL Central*.

The plug-in files are found in the following location in your SQL Anywhere installation:

Plug-in	File name and location
SQL Anywhere 17	*%SQLANY17%*`\java\saplugin.jar`
MobiLink 17	*%SQLANY17%*`\java\mlplugin.jar`
UltraLite 17	*%SQLANY17%*`\java\ulplugin.jar`

In this section:

1.7.2.4.1 Registering a Plug-in

Register an additional plug-in by specifying the plug-in registration file or the plug-in JAR file.

Procedure

1. In SQL Central, choose *Tools* > *Plug-ins*.

   The *SQL Central Plug-ins* window appears.
2. Select the plug-in you want to register, and then click *Register*.

   The *Register a Plug-In Wizard* appears.
3. Follow the instructions in the wizard.

Results

The plug-in is registered.

1.7.2.5 *SQL Anywhere 17* Plug-in

You can use the *SQL Anywhere 17* plug-in to upgrade existing databases, create new databases, and administer databases.

You can choose the mode from the *Mode* menu or by clicking the toolbar button for the mode.

The *SQL Anywhere 17* plug-in can operate in any of the following modes:

Design mode

While working in *Design* mode, you can create and modify database objects such as tables, users, triggers, indexes, remote database servers, and so on. You can also add data to tables, create new databases, and upgrade existing databases.

Debug mode

While working in *Debug* mode, you can use the SQL Anywhere debugger to assist you in developing SQL stored procedures, triggers, and event handlers.

Viewing SQL statements and utility commands generated by wizards

The final page of most *SQL Anywhere* plug-in wizards displays the SQL statements and utility commands that the wizards execute based on your choices.

Clicking *Finish* executes the SQL statements and/or runs the utility commands. Alternatively, you can copy the SQL statement to the clipboard, click *Cancel* to exit the wizard, and then execute the SQL statements via Interactive SQL.

This feature allows you to use the wizards to generate SQL scripts without modifying the database.

In this section:

Generate an HTML file that shows the dependencies and references for a database by using the *Database Documentation Wizard* in SQL Central.

Creating User-defined Folders (SQL Central) [page 1111]
Create user-defined folders to organize database objects. You can choose to maintain the folders manually or to have their contents update in response to changes in the database.

Related Information

Tables, Views, and Indexes
Tables, Views, and Indexes
The SQL Anywhere Debugger
SQL Anywhere Profiler [page 1483]

1.7.2.5.1 Comparing Database Schemas

Use SQL Central to compare two databases.

Prerequisites

You must have the SELECT ANY TABLE system privilege in both databases.

You must also have the ACCESS USER PASSWORD system privilege in both databases to compare their passwords.

Context

When comparing databases, objects are not matched by object ID but by name, name and owner, or a combination of names and owners.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the two databases you want to compare.
2. Click *Tools* > *SQL Anywhere 17* > *Compare Database Schema*.

   The *Compare Database Schema* window appears.
3. Specify the two databases you want to compare in the *Database 1* and *Database 2* fields.
4. Click *Compare*.

The *Objects* tab lists the summary of the differences between the databases.

5. Click an item on the *Objects* tab to view the SQL statements for the object. A red background indicates SQL for objects that exist in *Database 1* only, a green background for objects that exist in *Database 2* only, and a blue background for objects that exist in both databases.

Results

The databases are compared.

Next Steps

Optionally, to convert the schema of *Database 1* to match the schema of *Database 2*, the SQL statements necessary for the conversion appear on the *SQL Script* tab.

Related Information

SQL Script Files
Interactive SQL [page 1029]

1.7.2.5.2 Converting a Database Schema to Match Another

Use SQL Central to convert a database schema to match another.

Prerequisites

You must have the SELECT ANY TABLE system privilege in both databases.

You must have the privileges necessary for each SQL statement that must be run to convert the schema of *Database 1* to match the schema of *Database 2*.

The SQL script that is generated contains the statements necessary to convert the schema of *Database 1* to match the schema of *Database 2*. It is recommended that you back up *Database 1* before running the script, and that you ensure there are no other connections to the database while the script is running.

Context

The following operations cannot be performed on *Database 1* using this feature, and therefore are not included in the list of SQL statements generated during the comparison. When one of these operations is required, a notification describing the operation appears at the beginning of the list of SQL statements that is generated.

- Altering the file name for a dbspace
- Altering a domain
- Altering the read-only setting for a remote server
- Altering the dbspace for a table
- Altering the COMMIT action and/or SHARE BY ALL setting for a global temporary table
- Altering the order of columns in a table or global temporary table
- Altering the synchronization type for a publication

When comparing databases, objects are not matched by object ID, but by name, name and owner, or a combination of names and owners. Any objects with names in *Database 1* that do not appear in *Database 2*, including renamed tables and columns, are dropped and recreated, which may result in the loss of data.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the two databases you want to compare.
2. Click *Tools* > *SQL Anywhere 17* > *Compare Databases*.

   The *Compare Databases* window appears.
3. Ensure that the database you want to convert the schema of is specified in the *Database 1* field, and the database you want to convert the schema to is specified in the *Database 2* field.
4. Click *Compare*.

   The *Objects* tab displays the list of objects in the databases. The *SQL Script* tab displays the generated SQL statements.
5. Click an item on the *Objects* tab to view the differences in the SQL statements for the object. A red background indicates SQL for objects that exist in *Database 1* only, a green background for objects that exist in *Database 2* only, and a blue background for objects that exist in both databases.
6. Click *Open In Interactive SQL* on the *SQL Script* tab to open Interactive SQL with the statements pasted in the *SQL Statements* pane. You can make further edits as required and execute the statements. Executing the SQL statements makes *Database 1* the same as *Database 2*.

Results

The schema of *Database 1* is converted to match the schema of *Database 2*.

Related Information

SQL Script Files
Interactive SQL [page 1029]

1.7.2.5.3 Viewing Entity-relationship (ER) Diagrams

View an entity-relationship diagram of the tables in a database using SQL Central.

Prerequisites

You must have PUBLIC system role.

Context

You can also import and export the layout of the ER diagrams.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to a database.
2. Select the database, and then click the *ER Diagram* tab in the right pane to see the diagram.
3. Click *File* > *Choose ER Diagram Tables*, and specify the tables to appear in the ER diagram.
4. Click *OK*.
5. (Optional) Change the tables that appear in the diagram by adjusting the filtering set for the database.

   The tables that appear in the diagram are subject to the filtering set for the database. The tables are filtered by their owners. Click *File* > *Configure Owner Filter*.
6. Arrange the objects in the diagram. Drag the objects to change the layout. The layout changes persist between SQL Central sessions. You can also import a layout, by clicking *File* > *Import Layout*.
7. (Optional) Double-click a table to see the column definitions for that table.

Results

The entity-relationship diagram is changed.

Next Steps

Export the ER diagram by clicking *File* > *Export Layout*.

Related Information

Database Creation [page 277]

1.7.2.5.4 Database Health and Statistics

In Design mode, the *Overview* tab provides a high-level view of the database server, its health, statistics and its features.

This tab contains the following components:

Database

Located in the top left corner, this pane displays general information about the database server.

To update the database server software click *Check For Updates*.

Features

Located in the left bottom corner, this pane provides a visual representation of the database and its products and features. Clicking a node in the diagram expands the accompanying section in the *Health And Statistics* pane on the right; clicking the node again collapses the section.

> **i** Note
>
> You must initiate the retrieval of MobiLink information; otherwise, these nodes appear as unknowns (grayed-out).

Health And Statistics

Located on the right, this pane displays statistics and information relating to the overall status of the database. The following collapsible panes are available:

Statistics

Displays general statistics, such as the number of pages read and written to disk. It displays a warning if there are any unscheduled requests. Click the warning to learn more.

Dbspaces

Displays a table listing all dbspaces. It displays a warning if a dbspace has less than 10% of free disk space remaining, or if a dbspace file cannot be found. Click the warning to learn more.

Transaction Logs

Displays information for the transaction log and the transaction log mirror, if applicable. This pane appears only when the database has a transaction log. It displays a warning if a transaction log file has less than 10% of free disk space remaining. Click the warning to learn more.

Connected Users

Displays connected user and transaction statistics. Shows a table of the top 5 transaction times, if there are any. It shows a table of all blocked connections, if there are any. Displays a warning if there are any blocked connections. Click the warning to learn more.

Database Mirroring

Displays information for the primary, arbiter, mirror, and copy servers and for a database mirroring or read-only scale-out system when you are connected to the primary server for the system. This pane appears only when mirroring or scale-out is being used. It displays a warning if the arbiter or mirror server is disconnected. Click the warning to learn more.

Remote Servers

Displays a table of the remote servers used by the database. This pane appears only when a remote server exists. It displays a warning if a remote server is disconnected or a remote server cannot establish a connection. Click the warning to learn more.

MobiLink And Notifiers

Displays statistics for MobiLink and Notifiers. This pane only appears when MobiLink tables and views exist in the database.

> **i** Note
>
> You must click the *Refresh* button to refresh the information in this pane. Unlike the information in the other panes, the information in this pane is not refreshed when you click *View* > *Refresh* (or click the *Refresh* toolbar icon). You must refresh this information separately because refreshing could affect the database's performance.

SQL Remote Users

Displays a table of all SQL Remote users, and their most recent send and receive times. This pane only appears when the database has SQL Remote users.

Related Information

Software Updates [page 1113]

1.7.2.5.5 Documenting a Database (SQL Central)

Generate an HTML file that shows the dependencies and references for a database by using the *Database Documentation Wizard* in SQL Central.

Procedure

1. In SQL Central, click *Tools* > *SQL Anywhere 17* > *Generate Database Documentation*.
2. Follow the instructions in the wizard.

Results

The generated documentation is saved to HTML files, which makes it easy to navigate and review.

1.7.2.5.6 Creating User-defined Folders (SQL Central)

Create user-defined folders to organize database objects. You can choose to maintain the folders manually or to have their contents update in response to changes in the database.

Context

A folder holds one type of database object. You can create the following types of folders:

Folders

You determine the set of objects that the folder contains by choosing the objects in the database. Once the folder is created, you can add more objects to it by copying-and-pasting or dragging-and-dropping objects onto the folder. Objects in a plain folders remain in the folder until they are either explicitly removed from the folder or deleted from the database.

Smart folders You determine the set of objects that a smart folder contains by defining an expression that matches zero or more objects in the database. As you create, modify, and delete database objects, the contents of these smart folders change so that the folder always contains only those objects that match the folder's expression. Objects cannot be explicitly added to or removed from a smart folder.

The definitions for these user-defined folders are stored outside of the database and they are specific to the name of the database, database server, and name of the computer, which you used to create the folders. You can share your folder definitions by exporting them to a file and then importing them for use with a different database, database server.

Procedure

1. In SQL Central, connect to a database.
2. In the left pane, right-click the database and click *New* > *Folder*.
3. Type a folder name. Click *Next*.
4. Choose a folder type. Click *Next*.
5. Choose an object type. Click *Next*.
6. Choose one of the following options:

Option	Action
Folder	Specify the contents of the folder.

Option	Action
Smart folder	In the *Smart Folder Expression Editor*, define an expression that matches the database objects that you want in the folder.

7. Click *Finish*.

   You have created a folder. You cannot change the object type that the folder can contain. Create a new folder for that object type.
8. Share the folder with other users or with another database by exporting the folder definition to a file.
   a. In the left pane in, right-click the database, and then click *Export folders*.
   b. Follow the instructions in the wizard to select the folders to export. These folders can be imported for use with another database system or saved as a backup.

Results

You have created a definition file for your folders that you can use share with other users.

To import the folder definitions, connect to the different database system in SQL Centrall, and then right-click the database and choose *Import folders*.

Example

The following are sample expressions for smart folders:

- Show tables with Owner = 'DBA'

```
Owner = 'DBA'
```

- Show tables with names starting with 'APP1_'

```
Name STARTS WITH 'APP1_'
```

- Show tables with Owner 'DBA' and names starting with 'APP1_'

```
Owner = 'DBA' AND Name STARTS WITH 'APP1_'
```

- Show tables with Owner 'DBA' and names starting with 'APP1_' or containing '_test'

```
Owner = 'DBA' AND (Name STARTS WITH 'APP1_' OR Name CONTAINS '_test')
```

- Show tables with Owner 'DBA' and Type 'Proxy'

```
Owner = 'DBA' AND Type = 'Proxy'
```

- Show users with greater than five failed login attempts

```
FailedLoginAttempts > 5
```

- Show users with a last login time before '2013-01-01 00:00:00'

```
LastLoginTime < '2013-01-01 00:00:00'
```

- Show users with a last login time in the previous hour

```
LastLoginTime IN THE PREVIOUS '1h'
```

- Show users who have logged into the system

```
LastLoginTime IS NOT NULL
```

The following expression is for a smart folder containing *Users*. It displays user accounts that are unlocked, who have logged into the system,with names starting with the letter u, and who have more than 5 failed login attempts, a password that was created before noon on January 1, 2015, were last logged in the past 15 days.

```
LastLoginTime IS NOT NULL
AND Name STARTS WITH 'u'
AND FailedLoginAttempts > 5
AND Locked = 'No'
AND Type = 'User'
AND PasswordCreationTime < '2015-01-01 12:00:00'
AND LastLoginTime IN THE PREVIOUS '15d'
```

1.7.3 Software Updates

You can check for software updates in several ways.

Start menu

Click *Start* > *Programs* > *SQL Anywhere 17* > *Check For Updates*.

SQL Central

Click *Help* > *SQL Anywhere 17* > *Check For Updates*.

Interactive SQL

Click *Help* > *Check For Updates*.

SQL Anywhere Support utility (dbsupport)

Run the following command:

```
dbsupport -iu
```

In this section:

Configuring the Update Checker (SQL Central) [page 1114]
Configure the Update Checker to notify you when updates, such as Support Packages and minor releases, become available.

Configuring the Update Checker (Interactive SQL) [page 1115]
Configure the Update Checker to notify you when updates, such as Support Packages and minor releases, become available.

Related Information

Troubleshooting: Reporting an Error [page 1021]
A Guide to Downloading SAP SQL Anywhere Support Packages

1.7.3.1 Configuring the Update Checker (SQL Central)

Configure the Update Checker to notify you when updates, such as Support Packages and minor releases, become available.

Context

By default, SQL Anywhere checks daily for software updates.

Procedure

1. In SQL Central, click *Help* > *SQL Anywhere 17* > *Configure Update Checker*.
2. Edit the *Software Updates and Notices* settings.
3. Click *OK*.

Results

The Update Checker is configured.

Related Information

Support Utility (dbsupport) [page 1221]
SQL Anywhere Performance and Diagnostic Tracking

1.7.3.2 Configuring the Update Checker (Interactive SQL)

Configure the Update Checker to notify you when updates, such as Support Packages and minor releases, become available.

Context

By default, SQL Anywhere checks daily for software updates.

Procedure

1. In Interactive SQL, click *Tools* > *Options* > *Support*.
2. Edit the *Software Updates and Notices* settings.
3. Click *OK*.

Results

The Update Checker is configured.

Related Information

Support Utility (dbsupport) [page 1221]
SQL Anywhere Performance and Diagnostic Tracking

1.7.4 Database Administration Utilities

Several utility programs for performing database administration tasks are provided.

The administration utilities are used to perform a variety of management and maintenance functions.

In this section:

Configuration Files [page 1117]
Many of the utilities allow you to store command-line options in a configuration file.

Backup Utility (dbbackup) [page 1121]
Creates a client-side or a server-side backup of database files and transaction logs for running databases.

Broadcast Repeater Utility (dbns17) [page 1128]
Allows clients to find database servers running on other subnets and through firewalls where UDP broadcasts normally do not reach.

Certificate Creation Utility (createcert) [page 1130]
Creates X.509 certificates.

Certificate Viewer Utility (viewcert) [page 1138]
Displays values within a Public Key Infrastructure (PKI) object, converts the encoding of PKI objects, or encrypts and decrypts private keys.

Data Source Utility (dbdsn) [page 1140]
Creates, deletes, describes, and lists SQL Anywhere ODBC data sources.

dbisqlc Utility (Deprecated) [page 1154]
Executes SQL statements on a database.

Erase Utility (dberase) [page 1155]
Erases database files and associated transaction logs, or individual transaction log files.

Event Trace Data (ETD) File Management Utility (dbmanageetd) [page 1157]
Generates a diagnostic log that allows the user to examine information for user-defined and system events. Use the sp_trace_events system procedure to see the descriptions of the events.

File Hiding Utility (dbfhide) [page 1160]
Uses obfuscation or encryption to hide the contents of configuration files and initialization files.

Histogram Utility (dbhist) [page 1163]
Converts a histogram into a Microsoft Excel chart containing information about the selectivity of predicates.

Information Utility (dbinfo) [page 1165]
Displays information about the specified database.

Initialization Utility (dbinit) [page 1166]
Creates a new database.

Interactive SQL Utility (dbisql) [page 1179]
Executes SQL statements and runs script files against a database.

Key Pair Generator Utility (createkey) [page 1184]
Creates RSA key pairs for use with MobiLink end-to-end encryption.

Language Selection Utility (dblang) [page 1186]
Reports and changes the registry settings that control the language used by the database server, SQL Central, Interactive SQL, and other tools.

Log Translation Utility (dbtran) [page 1187]
Translates a transaction log into a SQL script file.

Performance Statistics Utility (dbstats) (UNIX/Linux) [page 1193]
Returns performance statistics for database servers on UNIX and Linux.

Ping Utility (dbping) [page 1195]
Tests connections to database servers and databases.

Profiler Utility (dbprof) [page 1199]
Starts the Profiler so that you can profile a database or review the contents of a previous profiling session.

Server Enumeration Utility (dblocate) [page 1200]
Locates database servers on the TCP/IP network.

Server Licensing Utility (dblic) [page 1203]
Applies your software license to your database server or MobiLink server.

Service Utility (dbsvc) for Linux [page 1207]
Creates, modifies, and deletes SQL Anywhere services.

Service Utility (dbsvc) for Windows [page 1212]
Creates, modifies, and deletes SQL Anywhere services.

Start Server in Background Utility (dbspawn) [page 1217]
Starts a database server in the background.

Stop Server Utility (dbstop) [page 1219]
Stops a database or database server.

Support Utility (dbsupport) [page 1221]
Sends performance data about errors and diagnostic information to Technical Support.

Transaction Log Utility (dblog) [page 1231]
Administers the transaction log for a database.

Unload Utility (dbunload) [page 1233]
Unloads a database into a specified directory.

Upgrade Utility (dbupgrad) [page 1254]
Updates the system tables and views, adds new database options, recreates all system stored procedures, installs jConnect support, archives the transaction log, and creates a new transaction log.

Validation Utility (dbvalid) [page 1257]
Validates the indexes and keys on tables and materialized views.

Version Diagnostic Utility (dbversion) [page 1262]
Returns information about the specified executable.

Related Information

Registry and INI Files [page 589]
SQL Central [page 1094]
Interactive SQL [page 1029]
SQL Anywhere Database Server Executable (dbsrv17, dbeng17) [page 380]

1.7.4.1 Configuration Files

Many of the utilities allow you to store command-line options in a configuration file.

If you use an extensive set of options, you may find it useful to store them in a configuration file.

The `@data` option allows you to specify an environment variable or configuration file on the command line.

To specify an environment variable, replace `@data` with the name of the environment variable.

To specify a configuration file, replace `data` with the path and name of the configuration file.

If both an environment variable and configuration file exist with the same name, the environment variable is used.

Configuration files can contain line breaks, and can contain any set of options, including the @data option. You can use the number sign (#) at the beginning of a line to designate the line as a comment. The ampersand (&) character appearing by itself at the end of a line indicates that the previous token is continued on the next line. For example, the following configuration file could be used to start a server that allows strong encryption:

```
-ec TLS(FIPS=Y;IDENTITY=rsaserver.id; &
     IDENTITY_PASSWORD=test)
 -x TCPIP c:\mydemo.db
```

The `@data` parameter can occur at any point in the command line, and parameters contained in the file are inserted at that point. You can use `@data` multiple times on one command line to specify multiple configuration files or environment variables.

Utilities read the command line by expanding the specified configuration files and reading the entire command line from left to right. If you specify options that are overridden by other options in the command line, the option closer to the end of the line wins. Conflicting options can cause errors.

> **i** Note
>
> The Start Server in Background utility (dbspawn) does not expand configuration files specified by the `@data` option.

To protect passwords or other information in a configuration file, you can use the File Hiding utility (dbfhide) to encrypt the contents of the configuration file.

Configuration File Escape Characters

Within a configuration file, if the value for an option contains values with spaces, then the value must be enclosed in double quotes. Furthermore, if the quoted value contains additional values with spaces, then the double quotes around the additional values must be escaped. SQL Anywhere supports \\ as an escape sequence for a \, and \" as an escape sequence for a ".

For example, this excerpt from a dblsn configuration file has escaped double quote characters:

```
-l "subject=$remote_id;
    content=sync cardealer;
    action='run dbmlsync.exe -c \"filedsn=c:\my fdsns\CarDealer.dsn\"
    -ot dbmlsync.log -k -e sa=on';"
```

Example

The following configuration file holds a set of options for the Validation utility (dbvalid):

```
#Connect to the sample database as the user DBA with password sql
-c "UID=DBA;PWD=sql;DBF=%SQLANYSAMP17%\demo.db"
```

```
#Perform an express check on each table
-fx
#Log output messages to the specified file
-o "c:\validationlog.txt"
```

If this configuration file is saved as `c:\config.txt`, it can be used in a command as follows:

```
dbvalid @c:\config.txt
```

In this section:

Related Information

File Hiding Utility (dbfhide) [page 1160]

1.7.4.1.1 Conditional Parsing in Configuration Files

You can use conditional parsing in configuration files to specify the utilities that can use the file.

Syntax

```
configuration-file= text...
```

```
text : comment | conditional | command-line-option
```

```
comment : line starting with # that is not a conditional
```

```
conditional :
```

```
#if  condition
text
   [ #elif  condition
text
   ] ...
   [ #else
   text
 ] ...
#endif
```

```
condition : { tool=utility-name[,utility-name]... | utility-name }
```

The following values are supported for `utility-name`:

dbbackup	dbinfo	dbmlsync	dbstop	dbxtract

dbdsn	dbinit	dbping	dbsupport	mlsrv
dbeng	dblic	dbremote	dbsvc	mluser
dberase	dblocate	dbspawn	dbunload	qaagent
dbfhide	dblog	dbsrv	dbupgrad	mlstop
dbhist	dblsn	dbstats	dbvalid	

Remarks

Conditional directives allow command parameters to be included or excluded depending on the utility using the file. The File Hiding utility (dbfhide) can still be used to encrypt the contents of a configuration file when conditional parsing is used in the file.

To be treated as a directive, the first non-whitespace character on a line must be #. When a utility is encountered in an #if or #elif directive, the lines that follow the directive are included until another conditional directive is encountered. The #else directive handles the condition where the utility has not been found in the preceding blocks. The #endif directive completes the conditional directive structure.

Blank spaces are not permitted anywhere within the list of tool names specified by *tool=*. You can nest conditional directives. If an error occurs while parsing the configuration file, the utility reports that the configuration file cannot be opened.

The dbspawn utility allows you to specify a configuration file reference in the command to be spawned, but you cannot specify a configuration file with options for the dbspawn utility. For example, the first command below is supported, but the second command is *not* supported:

```
dbspawn dbeng17 @myconfig.ini
dbspawn @spawnopts.ini dbeng17 demo.db
```

Example

The following configuration file can be used by dbping, dbstop, and dbvalid.

```
#if tool=dbping,dbstop,dbvalid
   #always make tools quiet
   -q
   -c "UID=DBA;PWD=passwd;Host=myhost;DBN=mydb"
   #if dbping
      #make a database connection
      -d
   #elif tool=dbstop
      #don't ask
      -y
   #else
      #must be dbvalid
      #use WITH EXPRESS CHECK
      -fx
   #endif
#endif
```

Related Information

Backup Utility (dbbackup) [page 1121]
Data Source Utility (dbdsn) [page 1140]
@data Database Server Option [page 397]
Erase Utility (dberase) [page 1155]
File Hiding Utility (dbfhide) [page 1160]
Histogram Utility (dbhist) [page 1163]
Information Utility (dbinfo) [page 1165]
Initialization Utility (dbinit) [page 1166]
Server Licensing Utility (dblic) [page 1203]
Server Enumeration Utility (dblocate) [page 1200]
Transaction Log Utility (dblog) [page 1231]
MobiLink Listener Utility for Windows (dblsn)
MobiLink SQL Anywhere Client Utility (dbmlsync) Syntax
Ping Utility (dbping) [page 1195]
SQL Remote Message Agent Utility (dbremote)
Start Server in Background Utility (dbspawn) [page 1217]
Stop Server Utility (dbstop) [page 1219]
Service Utility (dbsvc) for Linux [page 1207]
Service Utility (dbsvc) for Windows [page 1212]
Unload Utility (dbunload) [page 1233]
Upgrade Utility (dbupgrad) [page 1254]
Validation Utility (dbvalid) [page 1257]
Extraction Utility (dbxtract)
mlsrv17 Syntax
MobiLink User Authentication Utility (mluser)

1.7.4.2 Backup Utility (dbbackup)

Creates a client-side or a server-side backup of database files and transaction logs for running databases.

Syntax

```
dbbackup [ options ] target-directory
```

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.

Option	Description
-aw[-]	Enables automatic tuning of writers, which can improve overall backup performance by increasing the number of writers. Specifying -aw- disables the automatic tuning of writers and prevents the database server from creating additional writers. The -aw and -aw- options must be used with the -s option.
-b `block-size`	Specifies the maximum block size (in number of pages) to be used to transfer pages from the database server to dbbackup. The dbbackup utility tries to allocate this number of pages; if it fails, it repeatedly reduces this value by half until the allocation succeeds. The default size is 128 pages.
-bc `comment`	Records a comment in the backup history file. For archive backups, the comment is also recorded in the archive file. The -bc option must be used with the -s option.
-c `"keyword=value; ..."`	Specifies connection parameters.
-d	Backs up the main database files only, without backing up the transaction log file, if one exists.
-h[-]	Enables backup history, which appends a line to the `backup.syb` file. Specifying -h- disables backup history and prevents updates to the `backup.syb` file. The -h and -h- options must be used with the -s option.

Option	Description
-k `checkpoint-log-copy-option`	Specifies how dbbackup processes the database files before writing them to the destination directory. The choice of whether to apply pre-images during a backup, or copy the checkpoint log as part of the backup, has performance implications. If the -s option is specified to perform the backup on the server, the default setting for -k is auto; otherwise, the default setting is copy. **auto** The database server checks the amount of available disk space on the volume hosting the backup directory. If there is at least twice as much disk space available as the size of the database at the start of the backup, then the backup proceeds as if copy was specified. Otherwise, it proceeds as if nocopy was specified. This setting can only be used if -s is also specified. **copy** The backup reads the database files without applying pre-images for any modified pages. The checkpoint log in its entirety and the system dbspace are copied to the backup directory. The next time the database is started, the database server automatically recovers the database to its state as of the checkpoint at the start of the backup. Because page pre-images do not have to be written to the temporary file, using this option can provide better backup performance and reduce internal server contention for other connections that are operating during a backup. However, since the backup copy of the database file includes the checkpoint log, which has pre-images of any pages modified since the start of the backup, the backed-up copy of the database files may be larger than the database files at the time the backup started. The copy option should be used when disk space in the destination directory is not an issue. **nocopy** The checkpoint log is not copied as part of the backup. This option causes pre-images of modified pages to be saved in the temporary file so that they can be applied to the backup as it progresses. The backup copies of the database files will be the same size as the database when the backup operation commenced. The backup copies may actually be slightly

Option	Description
	smaller because the checkpoint log is not present in this copy. This option results in smaller backed up database files, but the backup may proceed more slowly, and possibly decrease performance of other operations in the database server. It is useful in situations where space on the destination drive is limited. **recover** The database server copies the checkpoint log (as with the copy option), but applies the checkpoint log to the database when the backup is complete. This restores the backed up database files to the same state (and size) that they were in at the start of the backup operation. This option is useful if space on the backup drive is limited (it requires the same amount of space as the copy option for backing up the checkpoint log, but the resulting file size is smaller). This setting can only be used if -s is also specified.
-l `filename`	Enables a secondary system to be brought up rapidly in the event of a server crash. A live backup does not stop. It continues running while the server runs. It runs until the primary server becomes unavailable. At that point, it shuts down, but the backed up log file is intact and can be used to bring a secondary system up quickly. If you specify -l, then you cannot use -s to create an image back up on the server.
-n	Changes the naming convention of the backup transaction log file to `yymmddxx.log`, where `xx` are sequential letters ranging from AA to ZZ and `yymmdd` represents the current year, month, and day. This option is used with -r. The backup copy of the transaction log file is stored in the directory specified in the command, and with the `yymmddxx.log` naming convention. This allows backups of multiple versions of the transaction log file to be kept in the same backup directory. You can also use both the -x option and the -n option to rename the log copy. For example: `dbbackup -c "UID=DBA;PWD=passwd" -x -n mybackupdir`
-o `filename`	Appends output messages to the named file.

Option	Description
-p	Sends formatted progress messages from the database server to the client. The -p option is ignored unless the -s option is used to perform a server-side backup.
-q	Suppresses output messages. This option is available only when you run this utility at a command prompt.
-r	Renames the transaction log and starts a new transaction log. It forces a checkpoint and causes the following three steps to occur: 1. The current working transaction log file is copied and saved to the directory specified in the command. 2. The current transaction log remains in its current directory, but is renamed using the format `yymmddxx.log`, where `xx` are sequential characters starting at `AA` and running through to `ZZ`, and `yymmdd` represents the current year, month, and day. This file is then no longer the current transaction log. 3. A new transaction log file is generated that contains no transactions. It is given the name of the file that was previously considered the current transaction log, and is used by the database server as the current transaction log.
-s	Creates an image backup on the server using the BACKUP DATABASE statement. If you specify the -s option, the -l option (to create a live backup of the transaction log) cannot be used. The directory specified is relative to the server's current directory so you must specify a full pathname. In addition, the server must have write permissions on the specified directory. When -s is specified, the Backup utility does not display progress messages and does not prompt you when it overwrites existing files. To be prompted when an attempt is made to overwrite an existing file, do not specify -s or -y. You must specify -s if you specify the -k recovery option.
-t	Creates a backup that can be used as an incremental backup since the transaction log can be applied to the most recently backed up copy of the database file(s).
-wa	Waits until transactions are complete to rename or truncate the transaction log. The -wa option is not supported by SQL Anywhere 16 or older servers unless the -s option is also used.

Option	Description
-wb	Delays the backup of the database until there are no active transactions. The -wb option must be used with the -s option.
-x	Backs up the existing transaction log, deletes the original log, and then starts a new transaction log. Do not use this option if you are using database mirroring.
-xo	Deletes the current transaction log and starts a new one. This operation does not perform a backup; its purpose is to free up disk space in non-replication environments. Do not use this option if you are using database mirroring. **⚠ Caution** Using this option can result in a database that cannot be recovered from media failure. Only use this option when data loss is acceptable. Do not use the -xo option with databases that are being replicated or synchronized. SQL Remote and MobiLink rely on transaction log information.
-y	Creates the backup directory or replaces a previous backup file in the directory without confirmation. To be prompted when an attempt is made to overwrite an existing file, do not specify -s or -y.
`target-directory`	Specifies the directory the backup files are copied to. If the directory does not exist, it is created. However, the parent directory must exist. By default, the Backup utility creates a client-side backup of the database files. You can specify -s to create a backup on the server using the BACKUP DATABASE statement.

Privileges

You must have the BACKUP DATABASE system privilege.

Remarks

The Backup utility makes a backup copy of all the files for a single database. A simple database consists of two files: the main database file and the transaction log. More complicated databases can store tables in multiple

files, with each file as a separate dbspace. All backup file names are the same as the database file names. The image backup created by the Backup utility consists of a separate file for each file that is backed up.

Using the Backup utility on a running database is equivalent to copying the database files when the database is not running. You can use the Backup utility to back up the database while other applications or users are using it.

If neither of the options -d or -t are used, all database files are backed up.

By default, the Backup utility creates a client-side backup of the database files. You can specify -s to create a backup on the server using the BACKUP DATABASE statement.

> ⚠ Caution
>
> Backup copies of the database and transaction log must not be changed in any way. If there were no transactions in progress during the backup, or if you specified BACKUP DATABASE WITH CHECKPOINT LOG RECOVER or WITH CHECKPOINT LOG NO COPY, you can check the validity of the backup database using read-only mode or by validating a copy of the backup database.
>
> However, if transactions were in progress, or if you specified BACKUP DATABASE WITH CHECKPOINT LOG COPY, the database server must perform recovery on the database when you start it. Recovery modifies the backup copy, which prevents subsequent transaction log files from the original database from being applied. In this case, you must use in-memory validation to prevent changing the database and transaction logs, or you can validate a copy of the database.

Exit codes are 0 (success) or non-zero (failure).

Example

For example, the following command backs up the sample database running on the computer named sample_host, connecting as the DBA user, into the `SQLAnybackup` directory:

```
dbbackup -c "Host=sample_host;DBN=demo;UID=DBA;PWD=sql" SQLAnybackup
```

Related Information

Transaction Log File Management in a Database Mirroring System [page 1775]
Database Backup and Recovery [page 939]
Types of Backup [page 941]
Alphabetical List of Connection Parameters [page 45]
Configuration Files [page 1117]
Backing up a Database Using the Backup Utility (dbbackup) [page 946]
Software Component Exit Codes
progress_messages Option [page 799]
BACKUP DATABASE Statement

1.7.4.3 Broadcast Repeater Utility (dbns17)

Allows clients to find database servers running on other subnets and through firewalls where UDP broadcasts normally do not reach.

Syntax

```
dbns17 [ options ] [ address ... ]
```

Option	Description
@data	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
-ap port	Specifies the port number used by the database server. The default port number is 2638.
-m ip	Specifies the IP address of the computer the DBNS process is running on. This parameter is required for computers with more than one IP address. This address must be an IPv4 address.
-o filename	Writes the output that appears in the Broadcast Repeater messages window to the named file.
-p port	Specifies the port number used by the DBNS Broadcast Repeater. The default is 3968. If there is a firewall between the subnets, then you must open the port number used by the Broadcast Repeater utility for TCP connections between DBNS processes, in addition to opening port 2638 for standard client-server communications.
-pf filename	Writes the process ID into the specified file.
-q	Runs in quiet mode--messages are not displayed.
-s	Causes the new DBNS process to check if another DBNS process is already running on that subnet, and returns an error before shutting down if another DBNS process is found.
-ud	(UNIX and Linux only) Runs the DBNS Broadcast Repeater as a daemon on UNIX and Linux operating systems. If you run the DBNS Broadcast Repeater as a daemon specify the -o option to log output information. When you start the DBNS Broadcast Repeater as a daemon, its permissions are controlled by the current user's umask setting. Set the umask value before starting the DBNS Broadcast Repeater to ensure that it has the appropriate permissions.

Option	Description
-ui	(UNIX and Linux only) For Linux with X window server support, starts the DBNS Broadcast Repeater in shell mode if a usable display is not available.
-ux	(UNIX and Linux only) Opens the *SQL Anywhere Broadcast Repeater* window on Linux (use the X window server).
-x `host`	Shuts down the DBNS process running on specified host. You can specify an IP address or host name.
-z	Starts the DBNS Broadcast Repeater in debug mode. When running in debug mode, a line appears in the Broadcast Repeater messages window for each SQL Anywhere broadcast packet that is received or forwarded. Debug mode should only be used when there are connectivity problems because of the verbosity of the debugging output.
`address`	Specifies the IP address or host name of other computers that are, or will be, running DBNS processes. This allows the DBNS processes to detect each other and exchange information about known database servers and other DBNS processes.

Privileges

None.

Remarks

The Broadcast Repeater allows clients to find SQL Anywhere database servers running on other subnets and through firewalls where UDP broadcasts normally do not reach, without using the Host connection parameter or LDAP.

The `address` can be either an IP address or a computer name. Use spaces to separate multiple addresses.

This utility is available on supported UNIX, Linux, and all 32-bit and 64-bit Microsoft Windows platforms.

The clients and database server must be running SQL Anywhere 9.0.2 or later to use the Broadcast Repeater.

> ⚠ Caution
>
> Do not run the dbns17 utility on the same computer as a SQL Anywhere database server because dbns17 or the database server may not receive UDP broadcasts.

Example

Suppose you want to allow computers on the subnets 10.50.83.255 and 10.50.125.255 to connect using broadcasts. You need to use a computer on the 10.50.83.255 subnet (Computer A at 10.50.83.114) and one computer on the 10.50.125.255 subnet (Computer B at 10.50.125.103).

On each of these two computers, run dbns17, passing the IP address of the other computer. Run the following command on Computer A:

```
dbns17 10.50.125.103
```

On Computer B, run the following command:

```
dbns17 10.50.83.114
```

If either computer has more than one IP address, you must specify the local IP address using the -m option.

On Computer A, run the following command:

```
dbns17 -m 10.50.83.114  10.50.125.103
```

Related Information

Troubleshooting: How the Broadcast Repeater Utility Locates Database Servers [page 266]
-pf Database Server Option [page 479]

1.7.4.4 Certificate Creation Utility (createcert)

Creates X.509 certificates.

Syntax

createcert [*-r* | *-s*]

Option	Prompt name	Description
@data	N/A	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, use the File Hiding utility (dbfhide) to encode the contents of the configuration file.

Option	Prompt name	Description
-b `value`	*Enter RSA key length (512-16384)*	Specifies the RSA key length. The length can between 512 bits and 16384 bits. This option is ignored when the -s option is specified.
-c `filename`	*Enter file path of signer's certificate*	Specifies a location and file name for the signer's certificate. If you supply this information, then the generated certificate is a signed certificate. If you do not supply this information, then the generated certificate is a self-signed root certificate. This option is ignored when the -r option is specified.
-ca {`0 | 1`}	*Certificate Authority (Y/N) [N]*	Creates a certificate authority certificate that can be used to sign other certificates. By default, certificates are not certificate authorities (0). This option is ignored when the -r option is specified.
-ck `filename`	*Enter file path of signer's private key*	A location and file name to save the private key associated with the certificate request. This option prompt only appears if you specify a file name for the -c option.
-co `filename`	*Enter file path to save certificate*	Writes the public certificate to the specified file. Specify a location and file name to save the certificate; otherwise, the certificate is not saved. This option is ignored when the -r option is specified.
-cp `password`	*Enter password for signer's private key*	Specifies the password that was used to encrypt the signer's private key. Only specify this password if the private key was encrypted. This option is ignored when the -r option is specified.

Option	Prompt name	Description
-io `filename`	*Enter file path to save identity*	Specifies a location and file name to save the generated identity. This option is ignored when the -r option is specified.
-ko `filename`	*Enter file path to save private key*	Specifies a location and file name to save the private key.
-kp `password`	*Enter password to protect private key*	Specifies a password with which to encrypt the private key. If you do not specify a password, the private key is not encrypted. This option only applies if you specify a file name with the -ko option.
-m `value`	*Serial number [generate GUID]*	Specifies a serial number. The serial number must be a hexadecimal string of 40 digits or less. This number must be unique among all certificates signed by the current signer. If you do not specify a serial number, createcert generates a GUID as the serial number. This option is ignored when the -r option is specified.
-p1	N/A	Uses PKCS #1 to encode unencrypted RSA private keys instead of the default, PKCS #8.
-r	N/A	Generates a PKCS #10 certificate request. When this option is specified, the following options do not apply: • -c • -ca • -ck • -co • -cp • -m • -io • -u • -x • -v

Option	Prompt name	Description
-ro `filename`	*Enter file path to save request*	Specifies a location and file name with which to save the PKCS #10 certificate request. This option only applies if you specify the -r option.
-s `filename`	N/A	Specifies the location and file name of the PKCS10 certificate request that is to be signed. The certificate request can be DER or PEM encoded. When this option is specified, the following options do not apply: • -b • -ec • -sc • -scn • -sl • -so • -sou • -sst • -t • -x
-sa `value`	N/A	Specifies which of the following signature algorithms to use: sha1, sha256, sha384, or sha512. The default algorithm is sha256.
-sc `value`	*Country Code*	Specifies the subject country code. This option is ignored when the -s option is specified.
-scn `value`	*Common Name*	Specifies the subject common name. This value must be the computer host name on which the certificate is to be used. It should match the Host=hostname that the client would use. This option is ignored when the -s option is specified.
-sl `value`	*Locality*	Specifies the subject locality. This option is ignored when the -s option is specified.

Option	Prompt name	Description
-so `value`	*Organization*	Specifies the subject organization. This option is ignored when the -s option is specified.
-sou `value`	*Organizational Unit*	Specifies the subject organizational unit. This option is ignored when the -s option is specified.
-sst `value`	*State/Province*	Specifies the subject state or province. This option is ignored when the -s option is specified.
-t `type`	N/A	Specifies the encryption type. Specify RSA. This option is ignored when the -s option is specified.

Option	Prompt name	Description
-u `value,...`	*Key usage*	Specifies how the certificate will be used. Specify a comma-separated list of numbers that indicate how the certificate's private key can be used. This option is an advanced option; the default should be acceptable for most situations. The default depends on whether the certificate is a certificate authority or not. The default for a certificate with certificate authority is *6,7*. The default for a certificate without certificate authority is *3,4,5*. This option is ignored when the -r option is specified. Values include: • 1. Digital Signature • 2. Nonrepudiation • 3. Key Encipherment • 4. Data Encipherment • 5. Key Agreement • 6. Certificate Signing • 7. CRL Signing • 8. Encipher Only • 9. Decipher Only
-v `years`	*Certificate valid for how many years (1-100)*	Specifies the number of years (between 1 and 100) that the certificate is valid. After this period, the certificate expires, along with all certificates it signs. This option is ignored when the -r option is specified.
-x	N/A	Generates a self-signed certificate. This option is ignored when the -r option is specified.

Option	Prompt name	Description
-3des	N/A	Encrypts the private key using 3DES encryption. Use this option to create a server identity file that can be used by older database servers running the Certicom encryption module as well as database servers that use the OpenSSL encryption module. Do not use this option if you are running in FIPS mode.

Privileges

None.

Remarks

Users typically go to a third party to purchase certificates. These certificate authorities provide their own tools for creating certificates. The following tools may be especially useful to create certificates for development and testing purposes, and can also be used for production certificates.

To create a signed certificate, you can specify no options and be prompted by createcert for the required values. Or, you can specify the required options for your certificate.

To break up the process into two steps, for example so one person creates a request and another person signs it, the first person can run createcert with -r to create a request and the second person can sign the request by running createcert with -s.

To create a root certificate (a certificate that signs other certificates), create a self-signed certificate with certificate authority and use the default signing options. For example, specify y for the *-ca* option and specify 6,7 for the *-u* option.

Private keys encrypted using that module were encrypted using 3DES encryption, instead of AES.

SQL Anywhere supports PKCS #1, PKCS #8 (default) and PKCS #10. If you use another format such as PKCS #12, convert the certificate into a supported format with an OpenSSL utility.

Example

The following example creates a self-signed certificate with the following characteristics:

- 1024-bit RSA key

- Generated serial number
- Valid for 5 years
- Not a certificate authority
- Used for digital signature, key encipherment, data encipherment, and key agreement
- Private key with the password sqlkey

```
createcert -t rsa -b 1024 -sc CA -sst ON -sl Waterloo -so MyCompany -sou MyUnit -
scn test -x -m 0 -v 5 -ca 0 -u 1,3,4,5 -co root-cert.pem -ko root-key.pem -io
root-id.pem -kp sqlkey
```

The following example creates a certificate signed by ca-key with the following characteristics:

- 1024-bit RSA key
- Generated serial number
- Valid for 5 years
- Not certificate authority
- Used for digital signature, key encipherment, data encipherment, and key agreement.
- Private key with the password sqlkey:

```
createcert -t rsa -b 1024 -sc CA -sst ON -sl Waterloo -so MyCompany -sou MyUnit -
scn test -c ca-cert.pem -ck ca-key.pem -cp cakeypass -m 0 -v 5 -ca 0 -u 1,3,4,5 -
co root-cert.pem -ko root-key.pem -io root-id.pem -kp sqlkey
```

The following example creates a signed certificate. In the example, because no file name is provided for the signer's certificate, the certificate is a self-signed root certificate.

```
createcert
SQL Anywhere X.509 Certificate Generator Version 17.0.11.1293
Warning: The certificate will not be compatible with older versions
of the software including version 12.0.1 prior to build 3994 and version 16.0
prior to build 1691. Use the -3des switch if you require compatibility.
Enter RSA key length (512-16384): 1024
Generating key pair...
Country Code: CA
State/Province: Ontario
Locality: Waterloo
Organization: MyCompany
Organizational Unit: Engineering
Common Name: Test Certificate
Enter file path of signer's certificate:
Certificate will be a self-signed root
Serial number [generate GUID]:
Generated serial number: 3ccd9e6b0c8c45d98f45f8a3f51d6ba0
Certificate valid for how many years (1-100): 10
Certificate Authority (Y/N) [N]: n
1.  Digital Signature
2.  Nonrepudiation
3.  Key Encipherment
4.  Data Encipherment
5.  Key Agreement
6.  Certificate Signing
7.  CRL Signing
8.  Encipher Only
9.  Decipher Only
Key Usage [1,3,4,5]: 1,3,4,5
Enter file path to save certificate: cert.pem
Enter file path to save private key: key.pem
Enter password to protect private key: pwd
Enter file path to save identity: id.pem
```

The following example creates a root certificate (a certificate that signs other certificates) that is a self-signed root certificate with Certificate Authority that uses the default signing. The response to the *Certificate Authority* prompt or the -ca option should be **y** and response to the Key usage prompt or the -u option should be **6,7** (the default).

```
Certificate Authority (Y/N) [N]: y
1.  Digital Signature
2.  Nonrepudiation
3.  Key Encipherment
4.  Data Encipherment
5.  Key Agreement
6.  Certificate Signing
7.  CRL Signing
8.  Encipher Only
9.  Decipher Only
Key Usage [6,7]: 6,7
```

Related Information

Transport Layer Security [page 1727]
FIPS-certified Encryption Technology [page 1730]
Certificate Viewer Utility (viewcert) [page 1138]
-ec Database Server Option [page 418]
Encryption (ENC) Connection Parameter [page 80]

1.7.4.5 Certificate Viewer Utility (viewcert)

Displays values within a Public Key Infrastructure (PKI) object, converts the encoding of PKI objects, or encrypts and decrypts private keys.

Syntax

viewcert [options] input-file

Option	Description
@data	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
-d	DER-encodes the output. This option is only useful with the -o option. It cannot be used with -p. By default, viewcert outputs the PKI object in a readable text format.

Option	Description
-ip `input-password`	Specifies the password needed to decrypt the private key if the `input-file` contains an encrypted private key.
-o `filename`	Specifies the file that viewcert should write the output to. By default, viewcert writes the output to the command prompt window where it is running.
-op `output-password`	Specifies the password viewcert should use to encrypt a private key. This option is only useful with -d or -p. By default, private keys are not encrypted.
-p	PEM-encodes the output. This option is only useful with the -o option. It cannot be used with -d. By default, viewcert outputs the PKI object in a readable text format.
-p1	Uses PKCS #1 to encode unencrypted RSA private keys instead of the default, PKCS #8.
-q	Runs in quiet mode--messages are not displayed.
`input-file`	Specifies a file that must be a DER- or PEM-encoded PKI object.
-3des	Specifies to use 3DES encryption instead of AES when specifying -p, -ip, and -op options. Do not use this option if you are running in FIPS mode.

Privileges

None.

Remarks

The viewcert utility can be used to view the following types of PKI objects:

- X.509 certificates
- certificate requests
- private keys
- certificate revocation lists (CRLs)

Viewcert can also be used to convert between DER and PEM encoding types and to encrypt or decrypt private keys.

The viewcert utility supports RSA objects.

Example

The following example allows you to view the sample RSA certificate that is included with SQL Anywhere. From the *%SQLANYSAMP17%*\Certificates directory, run the following command:

```
viewcert rsaroot.crt
```

This example produces the following output:

```
SQL Anywhere X.509 Certificate Viewer Version 17.0.0.1245
Common Name: Sample RSA Root Certificate
Country Code: CA
State/Province: ON
Locality: Waterloo
Organization: SAP
Organizational Unit: SQL Anywhere
Issuer: Sample RSA Root Certificate
Serial Number: 101
Issued: Apr 19, 2015 13:49:00
Expires: Apr 20, 2025 13:49:00
Signature Algorithm: RSA, SHA256
Key Type: RSA
Key Size: 2048 bits
Basic Constraints: Is a certificate authority, path length limit: 10
Key Usage: Certificate Signing, CRL Signing
```

Related Information

Separately Licensed Components
Certificate Creation Utility (createcert) [page 1130]

1.7.4.6 Data Source Utility (dbdsn)

Creates, deletes, describes, and lists SQL Anywhere ODBC data sources.

Syntax

```
dbdsn [ modifier-options ]
    { -l[ s | u ]
    | -d[ s | u ] dsn
    | -g[ s | u ] dsn
    | -w[ s | u ] dsn [details-options;...]
    | -cl }
```

Major option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
-l[*s* | *u*]	Lists the available ODBC data sources. You can modify the list format using the -b or -v options. On Microsoft Windows, you can modify the option using the *u* (user) or *s* (system) specifiers. The default specifier is *u*.
-d[*s* | *u*] `dsn`	Deletes the named data source. If you supply -y, any existing data source is deleted without confirmation. On Microsoft Windows, you can modify the option using the *u* (user) or *s* (system) specifiers. The default specifier is *u*.
-g[*s* | *u*] `dsn`	Lists the definition of the named data source. You can modify the format of the output using the -b or -v options. On Microsoft Windows, you can modify the option using the *u* (user) or *s* (system) specifiers. The default specifier is *u*.
-w[*s* | *u*] `dsn` [`details-options`]	Creates a new data source, or overwrites one if one of the same name exists. You must specify the -c option with the -w option. If you supply -y, any existing data source is overwritten without confirmation. On Microsoft Windows, you can modify the option using the *u* (user) or *s* (system) specifiers. The default specifier is *u*.
-cl	Lists the connection parameters supported by the dbdsn utility.

Modifier-options	Description
-b	Formats the output of the list as a single line connection string.

Modifier-options	Description
-cm	Displays the command used to create the data source. This option can be used to output the creation command to a file, which can be used to add the data source to another computer or can be used to restore a data source to its original state if changes have been made to it. You must specify the -g option or -l option with -cm or the command fails. Specifying -g displays the creation command for the specified data source, while specifying -l displays the creation command for all data sources. If the specified data source does not exist, the command to delete the data source is generated. For example, if the mydsn data source does not exist on the computer, `dbdsn -cm -g mydsn` would return the following command to delete the mydsn data source: `dbdsn -y -du "mydsn"`
-dr	Includes the Driver parameter when displaying data sources. When you use the -cm option to recreate data sources, it allows the current version of dbdsn to create data sources that reference a different version of the ODBC driver. For example, run the following command to create a version 11 data source: `dbdsn -y -wu "11.0 Student Sample" -c "UID=DBA;PWD=sql;...;Driver=SQL Anywhere 11"` When you run `dbdsn -cm -l`, dbdsn lists the same command without the Driver= parameter, which would then recreate the ODBC data source using the SQL Anywhere version 17 ODBC driver. However, if you run `dbdsn -dr -cm -l`, then the Driver= parameter is included and the data source is recreated exactly as it was created originally: using the version 11 ODBC driver.
-f	Displays the name of the system file that is being used. This option is only available on UNIX and Linux.

Modifier-options	Description
-ns	Specifies that the environment variable settings are used to determine the location of the system information file (named `.odbc.ini` by default). This option is also useful for determining which file is being used by dbdsn when there are multiple candidates for the system information file in the environment. This option is only available on UNIX and Linux. If you do not specify -ns when creating a data source, dbdsn also checks for the system information file in the user's home directory and the path.
-o `filename`	Appends output messages to the named file.
-or	Creates a data source for the SQL Anywhere 17 - Oracle driver when specified with the -c option. For example: `dbdsn -w MyOracleDSN -or -c Userid=DBA;Password=passwd; ServiceName=abcd;ArraySize=500;ProcResults=y` You can specify the -cl option with the -or option to obtain a list of the connection parameters for the SQL Anywhere 17 - Oracle driver.
-pe	Obfuscates the password specified in the Password (PWD) connection parameter, and stores the obfuscated password in the data source using the EncodedPassword (ENP) connection parameter. The obfuscated password can be used on any computer and provides no security. This option is not recommended and is deprecated.

Modifier-options	Description
-pet a | c | u	Encodes the password specified in the Password (PWD) connection parameter, and stores the encoded password in the data source using the ENP (EncodedPassword) connection parameter. **-pet a** The password is obfuscated for use on any computer. The encoded password string and its corresponding user ID can be used by anyone on any computer to initiate a connection to this database. **-pet c** The password is encoded on a computer such that it can only be decoded on that computer. The encoded password string and its corresponding user ID can only be used to authenticate to a database from the computer where the encoded password was created. Anyone who can log on to the computer can use the encoded password and its corresponding user ID to authenticate to a database. It cannot be used on any other computer. **-pet u** The password is encoded on a computer by a user such that it can only be decoded on that computer by that user only. The encoded password string and its corresponding user ID can only be used to authenticate to a database from the computer where the encoded password was created by the user who created it. It cannot be decoded by other users of that computer. It cannot be decoded on any other computer by the same or other user.
-q	Suppresses output to the database server messages window. If you specify -q when deleting or modifying a data source, you must also specify -y.
-v	Formats the output of the list over several lines, as a table.
-y	Deletes or overwrites each data source without prompting you for confirmation. If you specify -q when deleting or modifying a data source, you must also specify -y.

Details-options	Description
-c "keyword=value;..."	Specifies connection parameters as a connection string.
-cw	Ensures that the DBF parameter (specified using -c) is an absolute file name. If the value of DBF is not an absolute file name, dbdsn will prepend the current working directory (CWD). This option is useful because some operating systems do not have CWD information readily available in batch files.

Privileges

None.

Remarks

The modifier options can occur before or after the major option specification.

Any connection parameter that is stored in a DSN will not be used if the same parameter is given directly in the connection string or in the SQLCONNECT environment variable.

The Data Source utility is a cross-platform alternative to the ODBC Data Source Administrator for creating, deleting, describing, and listing ODBC data sources. The utility is useful for batch operations.

⚠ Caution

You should not store a password in a data source. Storing the password in plain text in an ODBC data source provides no security beyond the steps you take to prevent unauthorized access to the location on your computer where the ODBC data source is stored. It is best to prompt for a password and/or database key.

The -pet option stores an encoded password in a data source so that it is not in plain text. The ENP (EncodedPassword) field of the data source contains the encoded password as a hexadecimal string. The encoded string will be decoded back to plain text by the client software during a connect request. Note that this happens on the client computer before any connection attempt is made.

The -pet c and -pet u options can only be used on the computer or computer/user for which the encoded password is intended to be used.

Although -pet u can be used for a Microsoft Windows System data source, it may be more appropriate to use -pet c since this form of encoded password can be used by any user of the computer. If the system data source is created and used by only one Microsoft Windows user, then you can use -pet u.

i Note

On Microsoft Windows operating systems, data sources are stored in the registry or on disk. The Data Source utility does not support the use of file-based data sources using FILEDSN.

You cannot create a data source name that is longer than 32 characters.

i Note

On UNIX and Linux operating systems, data sources are held in the system information file (named `.odbc.ini` by default). When you use the Data Source utility to create or delete ODBC data sources on UNIX or Linux, the utility automatically updates the `[ODBC Data Sources]` section of the system information file. If you do not specify the Driver connection parameter using the -c option on UNIX or Linux, the Data Source utility automatically adds a Driver entry with the full path of the SQL Anywhere ODBC driver based on the setting of the SQLANY17 environment variable.

Do not encode the system information file (`.odbc.ini`) with the File Hiding utility (dbfhide) on UNIX or Linux unless you will only be using SQL Anywhere data sources. If you plan to use other data sources (for example, for MobiLink synchronization), then encoding the system information file may prevent other drivers from functioning properly.

Exit codes are 0 (success) or non-zero (failure).

Example

Write a definition of the user data source newdsn. If the data source already exists, a prompt to overwrite appears.

```
dbdsn -w newdsn -c "Host=myhost;Server=myserver;UID=DBA"
```

Write a definition of the user data source newdsn. If the data source already exists, do not prompt to overwrite.

```
dbdsn -y -w newdsn -c "Host=myhost;Server=myserver;UID=DBA"
```

Write a definition of the user data source newdsn containing an encoded form of the password. The data source can only be used by the user on the computer on which the data source was created.

```
dbdsn -pet u -w newdsn -c "Host=myhost;Server=myserver;UID=DBA;PWD=sql"
```

List all known user data sources, one data source name per line:

```
dbdsn -l
```

List all known system data sources, one data source name per line:

```
dbdsn -ls
```

List all user data sources along with their associated connection string:

```
dbdsn -l -b
```

Report the connection string for the user data source MyDSN:

```
dbdsn -g MyDSN
```

Report the connection string for the system data source MyDSN:

```
dbdsn -gs MyDSN
```

Delete the user data source BadDSN, but first list the connection parameters for BadDSN and prompt for confirmation:

```
dbdsn -d BadDSN -v
```

Delete the user data source BadDSN without prompting for confirmation.

```
dbdsn -d BadDSN -y
```

List all user data sources, logging output to `dsninfo.txt`:

```
dbdsn -l -o dsninfo.txt
```

List all connection parameter names and their aliases:

```
dbdsn -cl
```

List all connection parameter names, logging output to `dsninfo.txt`:

```
dbdsn -cl -o dsninfo.txt
```

Specify a relative file name. When the ODBC data source is created, the DatabaseFile connection parameter contains an absolute path such as `DBF=c:\SQLAnywhere17` if that was your current directory. For example:

```
c:\SQLAnywhere17> dbdsn -w testdsn -cw -c UID=DBA;Server=SQLAny;DBF=my.db
```

Run the following command to read the SQL Anywhere 17 Demo data source definition and output it to a file called `restoredsn.bat`:

```
dbdsn -cm -gs "SQL Anywhere 17 Demo" > restoredsn.bat
```

The `restoredsn.bat` file contains the following:

```
dbdsn -y -ws "SQL Anywhere 17 Demo" -c "UID=DBA;DBF='C:\Users\Public\Documents
\SQL Anywhere
         17\Samples\demo.db';
ServerName=demo17;START='C:\Program Files\SQL Anywhere 17\Bin64\dbeng17.exe';
ASTOP=YES;Description='SQL Anywhere 17 Sample Database'"
```

Run the following command to return the location of the system information file on UNIX or Linux:

```
dbdsn -f
```

This command returns the following output:

```
dbdsn using /home/user/.odbc.ini
```

Change the location of the system information file:

```
export ODBCINI=./myodbc.ini
```

Re-run `dbdsn -f` and the new location of the system information file is reported:

```
dbdsn using ./myodbc.ini
```

Use the -ns option when creating the data source to specify that the environment variable setting is used to determine the location of the system information file:

```
dbdsn -w NewDSN -c "UID=DBA;PWD=passwd" -ns
```

This results in the following output:

```
Configuration "newdsn" written to file ./myodbc.ini
```

In this section:

Related Information

Configuration Files [page 1117]
Alphabetical List of Connection Parameters [page 45]
SQL Anywhere 17 - Oracle ODBC Driver
ODBC Data Sources [page 116]
ODBC Data Sources on UNIX/Linux [page 126]
Creating an ODBC Data Source (ODBC Data Source Administrator) [page 117]
File Hiding Utility (dbfhide) [page 1160]
Software Component Exit Codes

1.7.4.6.1 ODBC Connection Parameters

ODBC connection parameters can be used to control the behavior of SQL Anywhere ODBC, JDBC, and OLE DB drivers.

ODBC connection parameters are specified along with other connection parameters in **connection strings**. In a connection string, a semicolon separates each connection parameter:

```
parameter1=value1;parameter2=value2;...
```

For example:

```
UID=DBA;PWD=passwd;DescribeCursor=always;IsolationLevel=2
```

Boolean (true or false) arguments are either YES or 1 if true, or NO or 0 if false.

These connection parameters are not supported by jConnect.

These connection parameters can be set in ODBC data sources using the Data Source utility (dbdsn) or the Microsoft ODBC Data Source Adminstrator.

Name	Description
AutoPreCommit	When the AutoPreCommit option is set to YES (AutoPre-Commit=yes), a commit is issued before each statement is executed, so that the application can always see the latest version of all database objects. The default value is NO.
ClientAutocommit	By default, automatic commits are handled by the database server for the ODBC, JDBC, and OLE DB application programming interfaces. This is called server-side autocommit. To disable server-side autocommit and enable client-side autocommit, set this option to YES (ClientAutocommit=yes). To restore the default behavior, set this option to NO (ClientAutocommit=no). This option is useful for frameworks that incur a tremendous performance penalty by constantly switching between manual commit and automatic commit.

Name	Description
Delphi	Delphi cannot handle multiple bookmark values for a row. When you set this value to NO, one bookmark value is assigned to each row, instead of the two that are otherwise assigned. Setting this option to YES can improve scrollable cursor performance.
DescribeCursor	This parameter lets you specify how often you want a cursor to be re-described when a procedure is called. The default setting is If Required. **Never** Specify 0, N, or NO if you know that your cursors do not have to be re-described. Re-describing cursors is expensive and can decrease performance. **If Required** Specify 1, Y, or YES if you want the ODBC driver to determine whether a cursor must be re-described. ODBC does not describe the result set after a cursor is opened. This is the default setting. **Always** If you specify 2, A, or ALWAYS, the cursor is re-described each time it is opened. If you use Transact-SQL procedures or procedures that return multiple result sets, you must re-describe the cursor each time it is opened.
Description	This parameter allows you to provide a description of the ODBC data source.
Driver	This parameter allows you to specify an ODBC driver for the connection, as follows: `Driver=driver-name`. By default, the driver that is used is SQL Anywhere 17. The `driver-name` must be SQL Anywhere `X`, where `X` is the major version number of the software. If the `driver-name` does not begin with *SQL Anywhere*, it cannot be read by the Data Source utility (dbdsn). On UNIX and Linux, this parameter specifies the fully qualified path to the shared object. If you do not specify the Driver connection parameter on UNIX or Linux, the Data Source utility (dbdsn) automatically adds a Driver entry with the full path of the SQL Anywhere ODBC driver based on the setting of the SQLANY17 environment variable.

Name	Description
Escape	This parameter specifies the escape character used in the LIKE clause of SQL statements that are generated by the ODBC driver when returning a list of tables or columns. By default the ODBC driver uses the tilde character (~), but some applications assume that the escape character is the backslash character (\). The following connection string fragment specifies that the backslash is the escape character: `"DSN=SQL Anywhere 17 Demo;UID=DBA;PWD=sql;ESCAPE=\ \;Host=myhost"`
GetTypeInfoChar	When this option is set to YES, CHAR columns are returned as SQL_CHAR instead of SQL_VARCHAR. By default, CHAR columns are returned as SQL_VARCHAR.
InitString	InitString allows you to specify a command that is run immediately after the connection is established. For example, you may want to set a database option or call a stored procedure.

Name	Description
IsolationLevel	You can specify one of the following values to set the initial isolation level for this data source: **0** This is also called the read uncommitted isolation level. This is the default isolation level. It provides the maximum level of concurrency, but dirty reads, non-repeatable reads, and phantom rows may be observed in result sets. **1** This is also called the read committed level. This provides less concurrency than level 0, but eliminates some of the inconsistencies in result sets at level 0. Non-repeatable rows and phantom rows may occur, but dirty reads are prevented. **2** This is also called the repeatable read level. Phantom rows may occur. Dirty reads and non-repeatable rows are prevented. **3** This is also called the serializable level. This provides the least concurrency, and is the strictest isolation level. Dirty reads, non-repeatable reads, and phantom rows are prevented. **Snapshot** You must enable snapshot isolation for the database to use this isolation level. The snapshot isolation levels prevent all interference between reads and writes. Writes can still interfere with each other. For contention, a few inconsistencies are possible and performance is the same as isolation level 0. **Statement-snapshot** You must enable snapshot isolation for the database to use this isolation level. The snapshot isolation levels prevent all interference between reads and writes. Writes can still interfere with each other. For contention, a few inconsistencies are possible and performance is the same as isolation level 0. **Readonly-statement-snapshot** You must enable snapshot isolation for the database to use this isolation level. The snapshot isolation levels prevent all interference between reads and writes. Writes can still interfere with each other. For contention,

Name	Description
	a few inconsistencies are possible and performance is the same as isolation level 0.
KeysInSQLStatistics, MSApplications, MSApp	Set this parameter to YES if you want the SQLStatistics function to return foreign keys. The ODBC specification states that SQLStatistics should not return primary and foreign keys; however, some Microsoft applications (such as Microsoft Visual Basic and Microsoft Access) assume that primary and foreign keys are returned by SQLStatistics. MSApplications and MSApp are synonyms for KeysInSQLStatistics.
LazyAutocommit	Set this parameter to YES to delay the commit operation until a statement closes.
MatView	Normally, a materialized view is reported as a VIEW by schema functions such as SQLTables. Set this parameter to some string value if you want to distinguish materialized views from views. For example, `MatView="MAT VIEW"` causes the ODBC driver to report a materialized view as MAT VIEW.
MaxLength	Use this parameter to set SQL_ATTR_MAX_LENGTH. The default value is 262144 (256K) . Since LONG VARCHAR, LONG NVARCHAR, and LONG BINARY columns can range as large as 2147483647 bytes, 256K was chosen as a compromise for a default buffer length by the ODBC driver. This default value can be overridden by setting SQL_ATTR_MAX_LENGTH with the SQLSetConnectAttr or SQLSetStmtAttr functions. An ODBC application can also handle long column values using SQLGetData calls to fetch the data in chunks. However, there are interfaces such as System.Data.Odbc that do not provide the ability to access these features. The connection string `"DSN=MyDataSource;DBN=demo;MaxLength=3145728"` sets SQL_ATTR_MAX_LENGTH to 3 MB. The MaxLength parameter can also be set in a Data Source using the Microsoft ODBC Data Source Administrator (ODBCAD32) or Data Source utility (dbdsn). In an ODBC application, the MaxLength parameter value, if it is specified, is overridden using SQL_ATTR_MAX_LENGTH with the SQLSetStmtAttr function.
PrefetchOnOpen	When PrefetchOnOpen is set to YES, a prefetch request is sent with a cursor open request. The prefetch eliminates a network request to fetch rows each time a cursor is opened. Columns must already be bound for the prefetch to occur on the open. This connection parameter can help reduce the number of client/server requests to help improve performance over a LAN or WAN.
PreventNotCapable	The SQL Anywhere ODBC driver returns an error because it does not support qualifiers. Some ODBC applications do not handle this error properly. Set this parameter to YES to prevent this error code from being returned, allowing these applications to work.

Name	Description
SkipRetryWithSingleRowMode	Set this parameter to YES to prevent the ODBC driver from retrying a failed batch operation in single-row mode. For example, a batch INSERT of 100,000 rows could fail somewhere on a primary key violation. By default, the driver would then attempt single-row inserts up to the point of failure. This can be time-consuming. This option can be used to prevent this, realizing that the point of the failure will be more difficult to determine.
SuppressInfoForDataTypes	Set this parameter to a comma-separated list of data type names to prevent the ODBC driver from returning information about the specified data types. The specified data types are not returned by the SQLGetTypeInfo or the DatabaseMetaData.getTypeInfo functions. But, the application can use the data type in column declarations and in result set queries. For example, the following connection-string fragment prevents the ODBC driver from returning information about the NVARCHAR, LONG VARCHAR, and VARBIT data types. `SuppressInfoForDataTypes=nvarchar,long nvarchar,varbit`
SuppressWarnings	Set this parameter to YES to suppress warning messages that are returned from the database server on a fetch. Versions 8.0.0 and later of the database server return a wider range of fetch warnings than earlier versions of the software. For applications that are deployed with an earlier version of the software, you can select this option to ensure that fetch warnings are handled properly.
TranslationDLL	This option is provided for backward compatibility. The use of translators is not recommended.
TranslationName	This option is provided for backward compatibility. The use of translators is not recommended.
TranslationOption	This option is provided for backward compatibility. The use of translators is not recommended.

Related Information

Guidelines for Choosing Isolation Levels

1.7.4.7 dbisqlc Utility (Deprecated)

Executes SQL statements on a database.

> **Note**
>
> The dbisqlc utility does not support all the SQL statements and features that Interactive SQL supports and may not support all the features available in the current version of the database server.Use the Interactive SQL utility.

Syntax

```
dbisqlc [ options ] [ dbisqlc-statement | dbisql-script-file ]
```

Option	Description
-c `"keyword=value;..."`	Specifies connection parameters. If Interactive SQL cannot connect, you are presented with a window where you can enter the connection parameters.
-d `delimiter`	Specifies a command delimiter. Quotation marks around the delimiter are optional, but are required when the command shell itself interprets the delimiter in some special way. The specified command delimiter is used for all connections in the current dbisqlc session.
-q	Suppresses output messages. This is useful only if you start Interactive SQL with a statement or script file. Specifying this option does not suppress error messages, but it does suppress the following: • warnings and other non-fatal messages • the printing of result sets
-x	Scans statements but does not execute them. This is useful for checking long script files for syntax errors.

Privileges

None.

Remarks

The dbisqlc utility allows you to type SQL statements or run SQL script files.

If `dbisqlc-statement` is specified, dbisqlc executes the statement. You can also specify a script file name. If no `dbisqlc-statement` or `dbisql-script-file` argument is specified, dbisqlc enters interactive mode, where you can type a statement into a command window.

Example

Run the following command to run the script file `mycom.sql` against the current default server, using the user ID DBA and the password passwd. If there is an error in the script file, the process shuts down.

```
dbisqlc -c "UID=DBA;PWD=passwd" mycom.sql
```

Run the following command to add a user to the current default database:

```
dbisqlc -c "UID=DBA;PWD=passwd" CREATE USER joe IDENTIFIED BY joespasswd
```

Related Information

Alphabetical List of Connection Parameters [page 45]
SQL Language Elements
Interactive SQL Utility (dbisql) [page 1179]

1.7.4.8 Erase Utility (dberase)

Erases database files and associated transaction logs, or individual transaction log files.

Syntax

dberase `[ options ] database-file`

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.

Option	Description
-ek `key`	Specifies the encryption key for strongly encrypted databases directly in the command. If you have a strongly encrypted database, you must provide the encryption key to use the database or transaction log in any way. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify the correct key for a strongly encrypted database.
-ep	Specifies that you want to be prompted for the encryption key. This option causes a window to appear, in which you enter the encryption key. It provides an extra measure of security by never allowing the encryption key to be seen in clear text. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify the correct key for a strongly encrypted database.
-o `filename`	Appends output messages to the named file.
-q	Runs in quiet mode--messages are not displayed. If you specify this option, you must also specify -y, otherwise the operation fails.
-y	Deletes each file without being prompted for confirmation. If you specify -q, you must also specify -y, otherwise the operation fails.

Privileges

None.

Remarks

With the Erase utility, you can erase a database file and its associated transaction log, or you can erase a transaction log file or transaction log mirror file. All database files and transaction log files are marked read-only to prevent accidental damage to the database and accidental deletion of the database files.

The `database-file` may be a database file or transaction log file. The full file name must be specified, including extension. If a database file is specified, the associated transaction log file (and mirror, if one is maintained) is also erased.

> **i** Note
>
> The Erase utility does not erase dbspaces. To erase a dbspace, use the DROP DBSPACE statement.

Deleting a database file that references other dbspaces does not automatically delete the dbspace files. To delete the dbspace files on your own, change the files from read-only to writable, and then delete the files

individually. Alternatively, you can use the DROP DATABASE statement or the *Erase Database Wizard* to erase a database and its associated dbspace files.

If you erase a database file, the associated transaction log and transaction log mirror are also deleted. If you erase a transaction log for a database that also maintains a transaction log mirror, the mirror is not deleted.

The database being erased must not be running when this utility is used.

Exit codes are 0 (success) or non-zero (failure).

Related Information

Configuration Files [page 1117]
Dropping a Database (dberase Utility) [page 314]
Dropping a Database (SQL Central) [page 313]
File Hiding Utility (dbfhide) [page 1160]
DROP DBSPACE Statement
DROP DATABASE Statement
Software Component Exit Codes

1.7.4.9 Event Trace Data (ETD) File Management Utility (dbmanageetd)

Generates a diagnostic log that allows the user to examine information for user-defined and system events. Use the sp_trace_events system procedure to see the descriptions of the events.

Syntax

```
dbmanageetd [ options ] [ filtering-options ] [ information-options ] [ output-format-options ] [ output-redirection-options ] filename [ filename2 ... ]
```

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
-m	Merges multiple diagnostic files.
-s	Shows a diagnostic summary. The -s option cannot be used with the -etd output format option.
-h	Shows the usage help.

Filtering option	Description
-fe `event-name`, ...	Filters by event name.
-fh `host`, ...	Filters by host name.
-fl `severity-level`	Filters by severity level. Valid severity levels are ALWAYS, CRITICAL, ERROR, WARNING, INFORMATION, or DEBUG. For example, WARNING include ALWAYS, CRITICAL, and ERROR but not INFORMATION or DEBUG.
-fr `offset`, `count`	Filters by record range. The offset is a zero-based offset. Count is the number of records returned.
-ft `from`, `to`	Filters by timestamp range. The timestamp format should be `YYYY-MM-DDThh:mm:ss.sssTZD` where `TZD` is the time zone designator (`Z` or +`hh:mm` or -`hh:mm`). Use `Z` for UTC offset or +`hh:mm` or -`hh:mm` to specify a time zone adjustment.
-fregex `regex`	Filters for the regular expression in event data (Windows and Linux only).

Information option	Description
-all	Shows all information (process ID, thread ID, host name, host information, and the file name).
-epoch	Shows the time in Epoch format, in milliseconds.
-filename	Shows the file name.
-hostinfo	Shows the host information. This information includes the operating system version and processor type.
-hostname	Shows the host name field.
-pid	Shows the process ID field.
-tid	Shows the thread ID field.
-tzadj `minutes`	Specifies the time zone adjustment value in minutes. The default is the local time of the computer running the utility.
-utc	Shows the time in UTC format.

Output format option	Description
-text	Shows output in the operating system character set. This value is the default. The operating system character set is used unless an encoding is specified with the -te option.
-etd	Shows output in ETD file format. Specify the -o option with this option. The -etd option cannot be used with the -s option.
-xml	Shows output in XML format using UTF-8.

Output format option	Description
-le	Lists available character set encodings. Specify + to view the expanded list.
-te `encoding`	Specifies the character set encoding for the -text output format. By default, the operating system character set is used.

Output redirection option	Description
-o `filename`	Specifies the name of the file where the output is saved.

Prerequisites

You must have already set up an event tracing session that logged information to an event trace data file.

Privileges

None.

Remarks

The -fe and -fh filtering options accept comma-delimited lists of values.

Output appears on the screen unless the -o option is specified.

Example

View a summary of the diagnostic information - Run the following command to view a summary of the diagnostic information:

```
dbmanageetd -s diagnostic-log.etd
```

View a list of events with a severity level of error and higher - Use the -fl option. For example:

```
dbmanageetd -fl error diagnostic-log
```

Convert an `.etd` file to an `.xml` file - Use the -xml and -o options. For example:

```
dbmanageetd -xml -o merged.xml diagnostic-log.etd
```

View only events on a specific host - To view only events from the host MyHost, use the -fh option. For example:

```
dbmanageetd -fh MyHost diagnostic-log.etd
```

Merge multiple `.etd` files from different hosts - Use the -m option. For example:

```
dbmanageetd -m -etd -o merged.etd machine1.etd machine2.etd machine3.etd
machine4.etd
```

View events filtered by timestamp range - To view events that occurred between 1:30 AM to 2:00 AM in EST on 26th of Mar 2012, use the -ft option. For example:

```
dbmanageetd -ft 2012-03-26T01:30:00.000-04:00,2012-03-26T02:00:00.000-04:00
diagnostic-log.etd
```

Related Information

Event Tracing [page 1012]
Viewing the Contents of the Event Trace Data (ETD) Diagnostic Log File [page 1018]

CREATE TEMPORARY TRACE EVENT Statement
CREATE TEMPORARY TRACE EVENT SESSION Statement
ALTER TRACE EVENT SESSION Statement
DROP TRACE EVENT Statement
DROP TRACE EVENT SESSION Statement
NOTIFY TRACE EVENT Statement
sp_read_etd System Procedure
sp_trace_events System Procedure
sp_trace_event_fields System Procedure
sp_trace_event_sessions System Procedure
sp_trace_event_session_events System Procedure
sp_trace_event_session_targets System Procedure
sp_trace_event_session_target_options System Procedure

1.7.4.10 File Hiding Utility (dbfhide)

Uses obfuscation or encryption to hide the contents of configuration files and initialization files.

Syntax

```
dbfhide [ options ] original-configuration-file encrypted-configuration-file
```

Option	Description
@data	Reads options from the specified environment variable or configuration file.

Option	Description
-q	Runs in quiet mode--messages are not displayed.
-w	Encodes the file for use by the current user on this computer only. The encoded file cannot be used by any other user on this computer, nor can it be used on any other computer.
-wm	Encodes the file for use by any user on this computer only. The encoded file cannot be used on any other computer.
`original-configuration-file`	Specifies the name of the original unencoded file.
`encrypted-configuration-file`	Specifies the name for the new encoded file.

Privileges

None.

Remarks

Some utilities use configuration files to hold command-line options.

By default, dbfhide uses simple obfuscation on the specified file. Simple obfuscation is intended only to keep data hidden in the event of casual direct access of the configuration file, to make it more difficult but not impossible for someone to decipher the contents of the configuration file.

On Microsoft Windows, the -w and -wm options use the Microsoft Cryptography API and the integrated Microsoft key store for strong encryption. The API links an encryption key to a specific user and computer (-w) or a specific computer (-wm).

On UNIX and Linux, the -w and -wm options use an obfuscation algorithm that is not secure.

When you specify the -w or -wm options, the file must be encoded on the target computer (for example, during a client software install). A file encoded on computer A with one of these options is not usable on computer B. If neither of the -w and -wm options is used, then the obfuscated file can be used on any computer.

⚠ Caution

It is recommended that passwords never be stored in a configuration file. Storing a password in a configuration file poses a security risk. If both user ID and password are discovered, then anyone with access to the database server can authenticate with that user ID and password. The contents of an encoded configuration file are secure only on Microsoft Windows when the -w or -wm option is used. The contents of an encoded configuration file are not secure on Linux, macOS, and other systems. Therefore, a configuration file should be secured with appropriate operating system permissions.

You cannot modify the contents of an encoded file using a text editor, and you cannot simply decode a file that has been encoded. Therefore you must preserve the original plain text file for future reference. To make

changes to an encoded file, you must edit the original plain text file and then use it to recreate the encoded file.

Example

Create a configuration file that names the database server `Elora`, sets the cache to 10 MB, and starts the sample database. The configuration file would be written as follows:

```
# Configuration file for server Elora
-n Elora
-c 10M
"C:\Users\Public\Documents\SQL Anywhere
         17\Samples\demo.db"
```

Lines beginning with # are treated as comments.

Name the file `sample.txt`. To start the sample database using this configuration file, enter:

```
dbsrv17 @sample.txt
```

Now, encode the configuration file.

```
dbfhide sample.txt sample.enc
```

Use the `sample.enc` file to start the sample database.

```
dbsrv17 @sample.enc
```

Encode the configuration file using options set in a Windows environment variable.

```
set hideopts=-q -wm
dbfhide @hideopts sample.txt sample.enc
```

Related Information

Configuration Files and Database Server Startup Options [page 336]
Data Security [page 1676]
Configuration Files [page 1117]
Hiding the Contents of an .ini File [page 591]
Conditional Parsing in Configuration Files [page 1119]

1.7.4.11 Histogram Utility (dbhist)

Converts a histogram into a Microsoft Excel chart containing information about the selectivity of predicates.

Syntax

```
dbhist [ options ] -t table-name [ excel-output-filename ]
```

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
-c `options`	Specifies connection parameters.
-n `colname`	Specifies the name of the column to associate the histogram with. If you do not specify a column, all columns that have histograms in the table are returned.
-t `table-name`	Specifies the name of the table or materialized view for which to generate the chart.
-u `owner`	Specifies the owner of the table or materialized view.
`excel-output-name`	Specifies the name of the generated Microsoft Excel file. If no name is specified, Microsoft Excel prompts you to enter one with a *Save As* window.

Privileges

You must have the MANAGE STATISTICS system privilege.

Remarks

Histograms are stored in the ISYSCOLSTAT system table and can also be retrieved with the sa_get_histogram stored procedure. The Histogram utility converts a histogram into a Microsoft Excel chart containing information about the selectivity of predicates. The Histogram utility (dbhist) only works on Windows, and you must have Microsoft Excel 97 or later installed.

Statistics (including histograms) may not be present for a table or materialized view, for example, if statistics were recently dropped. In this case, the Histogram utility returns the message `Histogram contains no data, aborting`. In this case, you must create the statistics, and then run the Histogram utility again. To create statistics for a table or materialized view, execute a CREATE STATISTICS statement.

To determine the selectivity of a predicate over a string column, use the ESTIMATE or ESTIMATE_SOURCE functions. Attempting to retrieve a histogram from string columns causes both sa_get_histogram and the Histogram utility to generate an error.

The sheets are named with the column name. Column names are truncated after 24 characters, and all occurrences of \, /, ?, *, [,], and : (which are not allowed in Microsoft Excel) are replaced with underscores (_). Chart names are prefixed with the word chart, followed by the same naming convention above. Duplicate names (arising from character replacement, truncation, or columns named starting with chart) result in a Microsoft Excel error stating that no duplicate names can be used. However, the spreadsheet is still created with those names created with their previous version (Sheet1, Chart1, and so on).

Exit codes are 0 (success) or non-zero (failure).

You can also retrieve histograms using the sa_get_histogram stored procedure.

Example

The following command (entered all on the same line) generates a Microsoft Excel chart for the column ProductID in the table SalesOrderItems for database *demo.db*, and saves it as `histogram.xls`.

```
dbhist -c "UID=DBA;PWD=sql;DBF=%SQLANYSAMP17%\demo.db" -n ProductID -t
SalesOrderItems histogram.xls
```

The following statement generates charts for every column with a histogram in the table SalesOrders, assuming that the sample database is already started. This statement also attempts to connect using UID=DBA and PWD=sql. No output file name is specified, so Microsoft Excel prompts you to enter one.

```
dbhist -t SalesOrders -c "UID=DBA;PWD=sql"
```

Related Information

Selectivity Estimate Sources
Optimizer Estimates and Statistics
Alphabetical List of Connection Parameters [page 45]
Configuration Files [page 1117]
CREATE STATISTICS Statement
Software Component Exit Codes
sa_get_histogram System Procedure
ESTIMATE Function [Miscellaneous]
ESTIMATE_SOURCE Function [Miscellaneous]

1.7.4.12 Information Utility (dbinfo)

Displays information about the specified database.

Syntax

```
dbinfo [ options ]
```

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
`-c "keyword=value;..."`	Specifies connection parameters.
`-o filename`	Appends output messages to the named file.
`-q`	Runs in quiet mode--messages are not displayed.
`-u`	Displays information about the usage and size of all tables, including system and user-defined tables and materialized views. You can only request page usage statistics if no other users are connected to the database. The page usage information is obtained using the sa_table_page_usage system procedure, which requires the MANAGE ANY STATISTICS privilege.

Privileges

Any user can run dbinfo. However, to obtain page usage statistics, you must have the MANAGE ANY DBSPACE system privilege.

Remarks

The dbinfo utility displays information about a database. It reports the name of the database file, the name of any transaction log file or log mirror, the page size, the collation name and label, whether table encryption is enabled, and other information. Optionally, it can also provide table usage statistics and details.

You can use the dbinfo utility to determine the size of a table on disk. To do so, run a command similar to the following:

```
dbinfo -u -c "UID=DBA;PWD=sql;DBF=%SQLANYSAMP17%\demo.db"
```

The result shows you how many pages are used to hold the data in each table in your database (Pages), and the percentage used of those pages (%used). For any table, you can then multiply the number of pages by the database page size, and then multiply that by %used to determine the amount of space is being used for your table.

You can also get more information about a database by:

- querying individual database properties using the DB_PROPERTY and DB_EXTENDED_PROPERTY system functions.
- querying all database properties using the sa_db_properties system procedure.
- using the -pd option with the Ping utility (dbping). When you specify -pd, dbping returns the value for each database property that is specified.

Exit codes are 0 (success) or non-zero (failure).

Related Information

Alphabetical List of Connection Parameters [page 45]
DB_PROPERTY Function [System]
sa_db_properties System Procedure
Ping Utility (dbping) [page 1195]
Software Component Exit Codes

1.7.4.13 Initialization Utility (dbinit)

Creates a new database.

Syntax

```
dbinit [ options ] new-database-file
```

Option	Description
@data	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, use the File Hiding utility (dbfhide) to encode the contents of the configuration file.

Option	Description
-a	Causes string comparisons to respect accent differences between letters (for example, e is less than é if the Unicode Collation Algorithm (UCA) is used for either CHAR or NCHAR data types (see -z and -zn). With the exception of Japanese databases created with a UCA collation, by default, accents are ignored (meaning e is equal to é). If all base letters (letters with accents and case removed) are otherwise equal, then accents are compared from left to right. The default accent sensitivity of a UCA collation when creating a Japanese database is *sensitive*. That is, accents are respected.
-af	Causes string comparisons to respect accent differences between letters (for example, e is less than é) if the UCA is used for either CHAR or NCHAR data types (see -z and -zn). By default, accents are ignored (meaning e is equal to é). If all base letters (letters with accents removed) are otherwise equal, then accents are compared from right to left, consistent with the rules of the French language.
-b	Blank pads the database. The database server compares all strings as if they are varying length and stored using the VARCHAR domain. This includes string comparisons involving fixed length CHAR or NCHAR columns. In addition, the database server never trims or pads values with trailing blanks when the values are stored in the database. By default, the database server treats blanks as significant characters. So, the value 'a ' (the character 'a' followed by a blank) is not equivalent to the single-character string 'a'. Inequality comparisons also treat a blank as any other character in the collation. If blank padding is enabled (the dbinit -b option), then the semantics of string comparisons more closely follow the ANSI/ISO SQL standard. With blank-padding enabled, the database server ignores trailing blanks in any comparison. In the example above, an equality comparison of 'a ' to 'a' in a blank-padded database returns TRUE. With a blank-padded database, fixed-length string values are padded with blanks when they are fetched by an application. The ansi_blanks connection option controls whether the application receives a string truncation warning on such an assignment.

Option	Description
-c	Considers all values case sensitive in comparisons and string operations. Identifiers in the database are case insensitive, even in case sensitive databases. With the exception of Japanese databases created with a UCA collation, the default behavior is that all comparisons are case *insensitive*. The default case sensitivity of a UCA collation when creating a Japanese database is *sensitive*. This option is provided for compatibility with the ISO/ANSI SQL standard.
-dba `DBA-user,pwd`	Required. Specifies a user ID and password that can be used to log in to the database after creation (also referred to as the DBA user). This user has full administrative rights on the database so that they can perform any privileged operation in the database and set up a security model. By default, passwords must be a minimum length of 6 characters unless the -mpl option is specified and set to a different value. Compose passwords with 7-bit ASCII characters. Other characters may not work correctly if the database server cannot convert from the client character set to UTF-8. The following command creates a database called mydb, with a DBA user called dba, that has the password passwd: `dbinit -dba dba,passwd mydb.db` The following command changes the DBA user to user1 with the password passwd: `dbinit -dba user1,passwd mydb.db`
-dbs `size`[*k* | *m* | *g* | *p*]	Preallocates space for the database. Preallocating space for the database helps reduce the risk of running out of space on the drive the database is located on. As well, it can help improve performance by increasing the amount of data that can be stored in the database before the database server needs to grow the database, which can be a time-consuming operation. By default, the `size` is in bytes. You can use *k*, *m*, *g*, or *p* to specify units of kilobytes, megabytes, or gigabytes, or pages, respectively.

Option	Description
-ea `algorithm`	Specifies the algorithm used for database or table encoding (-et). Specify `-ea simple` for simple obfuscation (do not specify -ek or -ep). Simple obfuscation is intended only to keep data hidden in the event of casual direct access of the database file, to make it more difficult for someone to decipher the data in your database using a disk utility to look at the file. Simple encryption is not secure. For greater security, specify one of AES, AES256, AES256CTR, or any of the FIPS-certified variants (for example, AES_FIPS, AES256CTR_FIPS). AES and AES_FIPS use 128-bit strong encryption while all the others are 256-bit strong. AES, AES256, and their FIPS variants encrypt with CBC block cipher mode. AES256CTR and AES256CTR_FIPS encrypt with CTR block cipher mode. Specify ARIA256 or ARIA256CTR for ARIA 256-bit encryption with CBC block cipher mode or CTR block cipher mode, respectively. ARIA is a variant of AES and considered of equivalent strength. For encryption aliases, see Encryption Algorithm Aliases [page 1701]. For strong encryption, you must also specify a key using the -ek or -ep option. To create a database that is not encrypted, specify `-ea none`, or do not include the -ea option (and do not specify -et, -ep, or -ek). If you do not specify the -ea option, the default behavior is as follows: • -ea NONE, if -ek, -ep, or -et is not specified • -ea AES, if -ek or -ep is specified (with or without -et) • -ea SIMPLE, if -et is used without -ek or -ep Algorithm names are case insensitive. The following command creates a strongly encrypted database and specifies the encryption key and algorithm. `dbinit -dba DBA,passwd -ek "0kZ2o56AK#" -ea AES_FIPS "myencrypteddb.db"` File compression utilities cannot compress encrypted database files as much as unencrypted ones.

Option	Description
-ek `key`	Specifies that you want to create a strongly encrypted database by specifying an encryption key directly in the command. The -ek option is used with an AES algorithm, optionally specified using the -ea option. If you specify the -ek option without specifying the -ea option, then AES is used by default. When specified with -et, the database is not encrypted. Instead, table encryption is enabled. **⚠ Caution** For strongly encrypted databases, store a copy of the key in a safe location. If you lose the encryption key, there is no way to access the data, even with the assistance of Technical Support. The database must be discarded and you must create a new database.
-ep	Specifies that you want to create a strongly encrypted database by inputting the encryption key in a window. This provides an extra measure of security by never allowing the encryption key to be seen in clear text. You must input the encryption key twice to confirm that it was entered correctly. If the keys don't match, the initialization fails. When specified with -et, the database is not encrypted. Instead, table encryption is enabled.

Option	Description
-et	Enables table encryption using the encryption algorithm (and key) specified for the -ea option. Use this option when you want to create encrypted tables instead of encrypting the entire database. If you specify -et with -ek or -ep, but not -ea, the AES algorithm is used by default. When you specify only -et, simple obfuscation is used. Enabling table encryption does not mean your tables are encrypted. You must encrypt tables individually, after database creation. When table encryption is enabled, table pages for the encrypted table, associated index pages, and temporary file pages are encrypted, and the transaction log pages that contain transactions on encrypted tables. The following example creates the database `new.db` with strong encryption enabled for tables using the key abc, and the AES_FIPS encryption algorithm: `dbinit -dba DBA,passwd -et -ek abc -ea AES_FIPS new.db`
-i	Excludes jConnect system objects from the database. To use the jConnect JDBC driver to access system catalog information, install jConnect catalog support (it is installed by default). When you specify this option you can still use JDBC, as long as you do not access system information. You can add jConnect support later using SQL Central or the ALTER DATABASE statement.
-k	Does not create the SYSCOLUMNS and SYSINDEXES views. By default, database creation generates the views SYS.SYSCOLUMNS and SYS.SYSINDEXES for compatibility with system tables that were available in Watcom SQL (versions 4 and earlier of this software). These views conflict with the SAP Adaptive Server Enterprise compatibility views dbo.syscolumns and dbo.sysindexes.

Option	Description
-kdi `number`	Specifies the number of times that the encryption key is hashed. Specify a whole number between 1 and 1000. The default value is 2, which results in the encryption key being hashed 2000 times. The higher the number, the better security. If you are running the database on a slow computer, then a very high number could result in the database taking longer to start. Use the -li option to estimate how this setting could affect the startup time (once the database is running, there is no performance impact). The -kdi option must be specified with either the -ek or the -ep option.
-l	Lists the available collation sequences and then stops. No database is created. To specify a collation sequence, use the -z option. To view a list of older collations, including alternate collations available for compatibility with older versions of SQL Anywhere and collations for special purposes, specify the -l+ option.
-le	Lists the available character set encodings and then stops. No database is created. Each character set encoding is identified by one or more labels. These are strings that can be used to identify the encoding. Each line of text that appears lists the encoding label and alternate labels by which the encoding can be identified. These labels fall into one of several common categories: SA (the SQL Anywhere label), IANA (Internet Assigned Numbers Authority), MIME (Multipurpose Internet Mail Extensions), ICU (International Components for Unicode), JAVA, or ASE (Adaptive Server Enterprise). To view a list of character set encodings that includes the alternate labels, specify the -le+ option. When the dbinit utility reports the character set encoding, it always reports the SQL Anywhere version of the label. For example, the following command reports the CHAR character set encoding windows-1250: `dbinit -dba DBA,passwd -ze cp1250 -zuca test.db`

Option	Description
-li `number`	Tests how long it takes your computer to create the encryption key based on the number of times that the encryption key is hashed. This time is equivalent to how long it can take for a brute force attack to try a password candidate. It also specifies the length of time that the software can take to validate the encryption key when the database starts. Use these results to guide you in choosing a value for the -kdi option. Specify a whole number between 1 and 1000. Do not specify other options when using the -li option.
-m `filename`	Creates a transaction log mirror. A transaction log mirror is an identical copy of a transaction log, usually maintained on a separate device, for greater protection of your data. By default, a transaction log mirror is not used.
-mpl `positive-integer`	Sets the minimum password length. If this option is not specified, then the default minimum password length for a new database is 6. This option does not specify the minimum utility database password length, which is always 6.
-n	Creates a database without a transaction log. Creating a database without a transaction log saves disk space, but can result in poorer performance because each commit causes a checkpoint. Also, if your database becomes corrupted and you are not running with a transaction log, data is not recoverable. The transaction log is required for data replication and provides extra security for database information during a media or system failure.
-o `filename`	Appends output messages to the named file.

Option	Description
-p `page-size` [*k*]	Specifies the page size for the database. The page size for a database can be (in bytes) 2048 (deprecated), 4096, 8192, 16384, or 32768, with 4096 being the default. Use *k* to specify units of kilobytes (for example, -p 4k). Large databases can benefit from a larger page size. For example, the number of I/O operations required to scan a table is generally lower, as a whole page is read in at a time. However, there are additional memory requirements for large page sizes. It is strongly recommended that you do performance testing (and testing in general) when choosing a page size. Then choose the smallest page size that gives satisfactory results. For most applications, 16 KB or 32 KB page sizes are not recommended. Do not use page sizes of 16 KB or 32 KB in production systems unless you can be sure that a large database server cache is always available, and only after you have investigated the trade offs of memory and disk space with its performance characteristics. If a large number of databases are going to be started on the same server, pick a reasonable page size.
-pd	Specifies that pre-16.0 system procedures that perform privileged operations will run with the privileges of the definer (owner), not the invoker. When -pd is specified, no additional privileges are required by a user to execute these system procedures other than EXECUTE privilege on the procedures. This setting does not impact system procedures added in version 16.0 or later that perform privileged operations, and does not impact the default behavior for user-defined procedures, which continues to be definer.
-q	Runs in quiet mode--messages are not displayed.

Option	Description
-s[-]	Adds global checksums (a checksum is added to each database page). By default, this option is on. To turn this option off, specify -s-. Checksums are used to determine whether a database page has been modified on disk. When you create a database with global checksums enabled, a checksum is calculated for each page just before it is written to disk. The next time the page is read from disk, the page's checksum is recalculated and compared to the checksum stored on the page. If the checksums are different, then the page has been modified or corrupted on disk, and an error occurs. Critical database pages are always checksummed by the database server, regardless of the value of the -s option. Checksums are automatically enabled for databases running on storage devices such as removable drives to help provide early detection if the database becomes corrupt. If a database is created with global checksums disabled (that is -s- is specified), you can still add checksums to pages when they are written by using the -wc option or the START DATABASE statement.
-*t* `transaction-log-name`	Specifies the name of the transaction log file. The transaction log is a file where the database server logs all changes, made by all users, no matter what application is being used. The transaction log plays a key role in backup and recovery, and in data replication. If the file name has no path, it is placed in the same directory as the database file. If you run dbinit without specifying -t or -n, a transaction log is created with the same file name as the database file, but with extension `.log`.

Option	Description
-z coll [collation-tailoring-string]	Specifies the collation sequence for the database. The collation sequence is used for sorting and comparing character data types (CHAR, VARCHAR, and LONG VARCHAR). The collation provides character comparison and ordering information for the encoding (character set) being used. It is important to choose your collation carefully. It cannot be changed after the database has been created without unloading and reloading the database. If the collation is not specified, the database server chooses a collation based on the operating system language and character set. To view the available collation sequences, see the -l option. Optionally, you can specify collation tailoring options (collation-tailoring-string) for additional control over the sorting and comparing of characters. These options take the form of keyword=value pairs, assembled in parentheses, following the collation name. For example: `dbinit -dba DBA,passwd -c -z uca(locale=es;case=LowerFirst) spanish2.db` Case and accent settings specified in the collation-tailoring-string override case and accent options for dbinit (-c, -a, and -af), if you specify both. **i Note** Databases initialized with collation tailoring options cannot be started by a pre-10.0.1 database server.
-ze encoding	Specifies the encoding for the collation. Most collations specified by -z dictate both the encoding (character set) and ordering. For those collations, -ze should not be specified. If the collation specified by -z is Unicode Collation Algorithm (UCA), then -ze can specify UTF-8 or any single-byte encoding for CHAR data types. By default, the database server uses UTF-8. Use -ze to specify a locale-specific encoding and get the benefits of the UCA for comparison and ordering.

Option	Description
-zn `coll` [`collation-tailoring-string`]	Specifies the collation sequence used for sorting and comparing of national character data types (NCHAR, NVARCHAR, and LONG NVARCHAR). The collation provides character ordering information for the UTF-8 encoding (character set) being used. Values are UCA (the default), or UTF8BIN which provides a binary ordering of all characters whose encoding is greater than 0x7E. If the dbicu17 and dbicudt17 DLLs are not installed, then the default NCHAR collation is UTF8BIN. Optionally, you can specify collation tailoring options (`collation-tailoring-string`) for additional control over the sorting and comparing of characters. These options take the form of keyword=value pairs, assembled in parentheses, following the collation name. For example: `dbinit -dba DBA,passwd -c -zn UCA(case=LowerFirst) sens.db` Case and accent settings specified in the `collation-tailoring-string` override case and accent options for dbinit (-c, -a, and -af), if you specify both. **i Note** Databases initialized with collation tailoring options cannot be started by a pre-10.0.1 database server.

Privileges

None.

Remarks

Several database attributes are specified at initialization and cannot be changed later except by unloading, reinitializing, and rebuilding the entire database. These database attributes include:

- Case sensitivity or insensitivity
- Accent sensitivity or insensitivity
- Punctuation sensitivity
- Treatment of trailing blanks in comparisons
- Page size

- Character set encoding or collation sequence
- Database encryption
- Table encryption

For example, the database `test.db` can be created with 8192 byte pages as follows:

```
dbinit -dba DBA,passwd -p 8192 test.db
```

Messages sent to the client indicate what type of database encryption is used for the database. If encryption is used, the algorithm being used is also displayed.

You cannot name a database utility_db. This name is reserved for the utility database.

When specifying collation tailoring options in the initialization command, you cannot specify quaternary for the punctuation sensitivity if the database is case or accent insensitive.

In addition, the choice of whether to use a transaction log and a transaction log mirror is made at initialization. This choice can be changed later using the Transaction Log utility or the ALTER DATABASE statement.

When encrypting the database or enabling table encryption in the database, specify an encryption key. The software uses Password-Based Key Derivation Function #2 (PBKDF2), which is part of the PKCS#5 standard to protect the key from brute-force attacks. The software repeatedly applies a cryptographic hash to the encryption key. Use the -kdi option to specify the number of times to apply the hash..

Exit codes are 0 (success) or non-zero (failure).

> **i** Note
>
> When you are deploying applications, the personal database server (dbeng17) is required for creating databases using the dbinit utility. It is also required if you are creating databases from SQL Central on the local computer when no other database servers are running.

Example

The following command creates a case sensitive database, `spanish.db`, which uses the 1252spa collation for non-NCHAR data. For NCHAR data, the UCA collation is specified, with locale es, and sorting by lowercase first.

```
dbinit -dba DBA,passwd -c -z 1252spa -zn uca(locale=es;case=LowerFirst)
spanish.db
```

The following command creates a database with a Greek collation.

```
dbinit -dba DBA,passwd -z 1253ELL mydb.db
```

Related Information

Unicode Collation Algorithm (UCA) [page 622]
Collation Considerations [page 627]
The Transaction Log [page 292]

Table Encryption [page 1706]
Simple Obfuscation Versus Strong Encryption [page 1698]
Procedures and Functions Running with Owner or Invoker Privileges
Running Pre-16.0 System Procedures as Invoker or Definer
Creating a Database (SQL Central) [page 278]
Encrypting a Table [page 1711]
Installing jConnect System Objects into a Database
ansi_blanks Option [page 677]
Recommended Character Sets and Collations [page 634]

1.7.4.14 Interactive SQL Utility (dbisql)

Executes SQL statements and runs script files against a database.

Syntax

```
dbisql -c "connection-string" [ options ] [ dbisql-statement | dbisql-script-file ]
```

```
dbisql -c "connection-string" -ul [ options ] [ dbisql-statement | dbisql-script-file ]
```

```
dbisql-statement: A SQL statement  or a series of sql statements separated by a command-delimiter.
```

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. If both the environment variable and configuration file exist with the same name, the environment variable is used. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to obfuscate the contents of the configuration file. Interactive SQL does not support configuration files that are encrypted.
`-c "keyword=value; ..."`	Specifies connection parameters. If Interactive SQL cannot connect, you are presented with a window where you can enter the connection parameters.
`-d delimiter`	Specifies a command delimiter. Quotation marks around the delimiter are optional, but are required when the command shell itself interprets the delimiter in some special way. This option overrides the setting of the Interactive SQL command_delimiter option.

Option	Description
-d1	Echoes all statements explicitly executed by the user to the command window (STDOUT). This can provide useful feedback for debugging SQL scripts, or when Interactive SQL is processing a long SQL script. (The final character is a number 1, not a lowercase L). This option is only available when you run Interactive SQL as a command line program.
-datasource `DSN-name`	Specifies an ODBC data source to connect to.
-f `filename`	Opens (but does not run) the file called `filename` in the SQL Statements pane. The file name can be enclosed in quotation marks, and *must* be enclosed in quotation marks if the file name contains a space. If the file does not exist, or if it is really a directory instead of a file, Interactive SQL prints an error message and then quits. If the file name does not include a full drive and path specification, it is assumed to be relative to the current directory. This option is only supported when Interactive SQL is run as a windowed application.
-hana	Specifies that you want to connect to an SAP HANA database by default. Interactive SQL customizes the options available to you depending on the type of database you are connected to. Regardless of the type of database set as the default, you can connect to SAP HANA, SQL Anywhere, SAP IQ, or UltraLite databases by choosing the database type from the *Change Database Type* dropdown list on the *Connect* window.
-host `hostname`	Specifies the `hostname` or IP address of the computer on which the database server is running. You can use the name localhost to represent the current computer.
-iq	Specifies that you want to connect to an SAP IQ database by default. Interactive SQL customizes the options available to you depending on the type of database you are connected to. Regardless of the type of database set as the default, you can connect to SAP HANA, SQL Anywhere, SAP IQ, or UltraLite databases by choosing the database type from the *Change Database Type* dropdown list on the *Connect* window.

Option	Description
-nogui	Runs Interactive SQL as a console application, with no windowed user interface. This is useful for batch operations. If you specify either `dbisql-statement` or `dbisql-script-file`, then -nogui is assumed. In this mode, Interactive SQL sets the program exit code to indicate success or failure. On Windows operating systems, the environment variable ERRORLEVEL is set to the program exit code.
-nohistory	Prevents the SQL history from being saved when Interactive SQL is terminated. Executed statements still appear on the *History* tab while the program is running.
-onerror { *continue* | *exit* }	Controls what happens if an error is encountered while reading statements from a script file. It is useful when using Interactive SQL in batch operations. This option overrides the Interactive SQL on_error option setting. Define one of the following supported `behavior` values: **Continue** The error is ignored and Interactive SQL continues executing statements. **Exit** Interactive SQL terminates.
-port `port-number`	Specifies the port number on which the SQL Anywhere database server is running. The default port number is 2638.[1]
-q	Suppresses output messages. Sets the utility to run in quiet mode. This is useful only if you start Interactive SQL with a statement or script file. Specifying this option does not suppress error messages, but it does suppress the following: • warnings and other non-fatal messages. When the -we option is specified, only non-fatal messages are suppressed. • the printing of result sets

Option	Description
-sa	Specifies that you want to connect to a SQL Anywhere database by default. Interactive SQL customizes the options available to you depending on the type of database you are connected to. Regardless of the type of database set as the default, you can connect to SAP HANA, SQL Anywhere, SAP IQ, or UltraLite databases by choosing the database type from the *Change Database Type* dropdown list on the *Connect* window.
-ul	Specifies that you want to connect to an UltraLite database by default. Interactive SQL customizes the options available to you depending on the type of database you are connected to. Regardless of the type of database set as the default, you can connect to SAP HANA, SQL Anywhere, SAP IQ, or UltraLite databases by choosing the database type from the *Change Database Type* dropdown list on the *Connect* window.
-version	Displays the version number of Interactive SQL. You can also view the version number from within Interactive SQL; from the *Help* menu, click *About Interactive SQL*.
-we	Displays and treats SQL statement warnings as though they are errors. As a result, when the -q option specified, warnings are not suppressed.
-x	Scans statements but does not execute them. This is useful for checking long script files for syntax errors.
`dbisql-statement` | `dbisql-script-file`	Execute the SQL statement or execute the specified `dbisql-script-file`. If you do not specify a `dbisql-statement` or `dbisql-script-file`, Interactive SQL enters interactive mode, where you can type a statement into a command window.

[1]This option is ignored for UltraLite.

Remarks

Interactive SQL allows you to browse the database, execute SQL statements, and run script files. It also provides feedback about:

- the number of rows affected
- the time required for each statement
- the execution plan of queries
- any error messages

You can use Interactive SQL to connect to a SQL Anywhere database, an UltraLite database, an SAP IQ database, an SAP HANA database, or a generic ODBC database.

For Windows, there are two executables:

1. Batch scripts should call `dbisql` or `dbisql.com`, not `dbisql.exe`. The `dbisql.com` executable is linked as a console application.
2. The `dbisql.exe` executable is linked as a windowed application and does not block the command shell from which it was started. If `dbisql.exe` is run from a batch file, you won't see any output sent to the standard output or standard error files.

The default encoding for Interactive SQL can also be temporarily set using the Interactive SQL default_isql_encoding option.

You can specify the encoding to use when reading or writing files using the ENCODING clause of the INPUT, OUTPUT, or READ statement.

- INPUT statement
- OUTPUT statement
- READ statement

Exit codes are 0 (success) or non-zero (failure). Non-zero exit codes are set only when you run Interactive SQL in batch mode (with a command line that contains a SQL statement or the name of a script file).

For UltraLite, when in command-prompt mode, Interactive SQL sets the program exit code to indicate success or failure. On Windows operating systems, the environment variable ERRORLEVEL is set to the program exit code.

When executing a `reload.sql` file with Interactive SQL and the database is encrypted, you must specify the encryption key as a parameter. If you do not provide the key in the READ statement, Interactive SQL prompts for the key.

You can start Interactive SQL in the following ways:

- From SQL Central, by clicking *File* > *Open Interactive SQL*.
- From the *Start* menu by clicking *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *Interactive SQL*.
- Using the dbisql command at a command prompt.

Example

The following command runs the script file `mycom.sql` against the current default server, using the user ID DBA and the password passwd. If there is an error in the script file, the process shuts down.

```
dbisql -c "UID=DBA;PWD=passwd" -onerror exit mycom.sql
```

The following command runs the script file `mycom.sql` against the `CustDB.udb` database for UltraLite. The -onerror option is defined as Exit; so, if there is an error in the script file, the process terminates.

```
dbisql -ul -c "UID=DBA;PWD=passwd;DBF=CustDB.udb" -onerror exit mycom.sql
```

The following command adds a user to the current default database:

```
dbisql -c "UID=DBA;PWD=passwd" CREATE USER joe IDENTIFIED BY joespasswd
```

The following command opens an SAP HANA database:

```
dbisql -hana -c "user=test;password=notSoSecret;host=akbar:30015"
```

Related Information

SQL Statements for Interactive SQL [page 1046]
Interactive SQL [page 1029]
SQL Statements for Interactive SQL [page 1046]
Configuration Files [page 1117]
Alphabetical List of Connection Parameters [page 45]
Database Creation [page 277]
File Hiding Utility (dbfhide) [page 1160]
INPUT Statement [Interactive SQL]
OUTPUT Statement [Interactive SQL]
READ Statement [Interactive SQL]
UltraLite Connection Parameters
Supported Exit Codes
command_delimiter Option [Interactive SQL] [page 1059]
default_isql_encoding Option [Interactive SQL] [page 1061]
on_error Option [Interactive SQL] [page 1072]

1.7.4.15 Key Pair Generator Utility (createkey)

Creates RSA key pairs for use with MobiLink end-to-end encryption.

Syntax

createkey

Remarks

When you run createkey, you are prompted for the following information:

Enter RSA key length (512-16384)

This prompt only appears if you chose RSA encryption. You can choose a length between 512 bits and 16384 bits.

Enter file path to save public key

Specify a file name and location for the generated PEM or DER encoded public key. This file is specified on the MobiLink client by the e2ee_public_key protocol option.

Enter file path to save private key

Specify a file name and location for the generated PEM or DER encoded private key. This file is specified on the MobiLink server via the e2ee_private_key protocol option.

Enter password to protect private key

Optionally, supply a password with which to encrypt the private key. The private key is not encrypted if you do not supply a password. This password is specified on the MobiLink server via the e2ee_private_key_password protocol option.

Privileges

None.

Example

The following example creates an RSA key pair:

```
createkey
SQL Anywhere Key Pair Generator Version 17.0.11.1293
Enter RSA key length (512-16384): 2048
Generating key pair...
Enter file path to save public key: rsapublic.pem
Enter file path to save private key: rsaprivate.pem
Enter password to protect private key: pwd
```

Related Information

End-to-end Encryption
e2ee_public_key MobiLink Client Network Protocol Option
-x mlsrv17 Option

1.7.4.16 Language Selection Utility (dblang)

Reports and changes the registry settings that control the language used by the database server, SQL Central, Interactive SQL, and other tools.

Syntax

```
dblang [ options ] language-code
```

Option	Description
-m	Writes the language code to the registry under HKEY_LOCAL_MACHINE.
-q	Runs in quiet mode--messages are not displayed.
-u	Writes the language code to the registry under HKEY_CURRENT_USER. This is the default location.

Language code	Language
EN	English
DE	German
ES	Spanish
FR	French
IT	Italian
JA	Japanese
KO	Korean
LT	Lithuanian
PL	Polish
PT	Portuguese
RU	Russian
TW	Traditional Chinese
UK	Ukrainian
ZH	Simplified Chinese

Privileges

None.

Remarks

If you do not specify -m or -u, then the language code is written to the registry under HKEY_CURRENT_USER. You can specify both -m and -u to write the language code to both locations.

Running the dblang utility without a language code reports the current settings.

> **i** Note
>
> Changes to language settings are only detected after restarting software components.

Exit codes are 0 (success) or non-zero (failure).

This utility does *not* accept the *@data* parameter to read in options from a configuration file.

Example

The following command displays a window containing the current settings:

```
dblang
```

The following command changes the settings to German, and displays a window containing the previous and new settings:

```
dblang de
```

Related Information

Software Component Exit Codes
SALANG Environment Variable [page 576]
Language (LANG) Connection Parameter [page 93]

1.7.4.17 Log Translation Utility (dbtran)

Translates a transaction log into a SQL script file.

> Syntax
>
> Running against a database server:
>
> ```
> dbtran [ options ] -c { connection-string } -n SQL-script-file
> ```
>
> Running against a transaction log:
>
> ```
> dbtran [ options ] [ transaction-log ] [ SQL-script-file ]
> ```

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
-a	Controls whether uncommitted transactions appear in the SQL script file. By default, the SQL script file only contains changes that have been committed. Changes that were rolled back or made after the most recent COMMIT are not present in the SQL script file. If -a is specified, any uncommitted transactions found in the transaction log are written to the SQL script followed by a ROLLBACK statement.
-c `"keyword=value;..."`	Specifies the connection string when running the utility against a database server.
-d	Specifies that transactions are written in order from earliest to latest. This feature makes the output easier to read and is provided primarily for use when auditing database activity. **⚠ Caution** Chronological ordered output must not be applied to a database because running statements in chronological order does not guarantee that the resulting database will be identical.

Option	Description
-ek `key`	Specifies the encryption key for strongly encrypted databases. If you have a strongly encrypted database, you must provide the encryption key to use the database or transaction log. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command fails if you do not specify the correct encryption key. If you are running dbtran against a database server using the -c option, specify the key using a connection parameter instead of using the -ek option. For example, the following command gets the transaction log information about database `myenc.db` from the database server **`sample`**, and saves its output in `log.sql`. `dbtran -n log.sql -c "Server=sample;DBF=myenc.db; UID=DBA;PWD=passwd;DBKEY=mykey"`
-ep	Prompts for the encryption key. This option causes a window to appear, in which you enter the encryption key. It provides an extra measure of security by never allowing the encryption key to be seen in clear text. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command fails if you do not specify the correct encryption key. If you are running dbtran against a database server using the -c option, specify the key using a connection parameter, instead of using the -ep option. For example, the following command gets the transaction log information about database `myenc.db` from the database server **`sample`**, and saves its output in `log.sql`. `dbtran -n log.sql -c "Server=sample;DBF=myenc.db; UID=DBA;PWD=passwd;DBKEY=mykey"`
-f	Outputs only transactions that were completed since the last checkpoint.
-g	Adds auditing information to the SQL script file (as comments) if the auditing database option is turned on. The -g option implies the -a, -d, and -t options.
-ir `offset1,offset2`	Outputs a portion of the transaction log between two specified offsets.

Option	Description
-is `source`,...	Outputs operations on rows that have been modified by operations from one or more of the following sources, specified as a comma-separated list: **All** All rows. This is the default setting. **SQLRemote** Include only rows that were modified using SQL Remote. You can also use the short form SR. **Local** Include only rows that are not replicated.
-it `owner.table`,...	Outputs those operations on the specified, comma-separated list of tables. Each table should be specified as `owner.table`.
-j `date/time`	Translates only transactions from the most recent checkpoint before the given date and/or time. The user-provided argument can be a date, time, or date and time, enclosed in quotes. If the time is omitted, the time is assumed to be the beginning of the day. If the date is omitted, the current day is assumed. The following is an acceptable format for the date and time: `"YYYY/MMM/DD HH:NN"`.
-k	Prevents partial `.sql` files from being erased if an error is detected. If an error is detected while dbtran is running, the `.sql` file generated until that point is normally erased to ensure that a partial file is not used by accident. Specifying this option may be useful if you are attempting to salvage transactions from a damaged transaction log.
-m `directory`	Specifies the directory that contains transaction logs. Use the -m option when transactions might span multiple transaction log files. When generating a SQL output file from multiple transaction logs, the Log Translation utility interleaves the operations from the logs into a sequence that reflects the order in which the operations would be applied when restoring the database. This option requires the -n option.
-n `filename`	Specifies the SQL output file.
-o `filename`	Appends output messages to the named file.
-q	Runs in quiet mode--messages are not displayed.
-r	Removes any transactions that were not committed. This is the default behavior.

Option	Description
-s	Controls how UPDATE statements are generated. If the option is not used, and there is no primary key or unique index on a table, the Log Translation utility generates UPDATE statements with a non-standard FIRST keyword if there are duplicate rows. If the option is used, the FIRST keyword is omitted for compatibility with the SQL standard.
-sr	Places generated comments in the output file describing how SQL Remote distributes operations to remote sites.
-t	Controls whether triggers are included in the script file. By default, actions performed by triggers are not included in the script file. If the matching trigger is in the database, when the script file is run against the database the trigger performs the actions automatically. Trigger actions should be included if the matching trigger does not exist in the database against which the script file is to run.
-u `userid`,...	Limits the output from the transaction log to include only specified users.
-x `userid`,...	Limits the output from the transaction log to exclude specified users.
-y	Replaces existing script files without prompting you for confirmation. If you specify -q, you must also specify -y or the operation fails.
-z	Includes transactions that were generated by triggers only as comments in the output file.
`transaction-log`	Specifies the transaction log file to be translated. Cannot be used together with -c or -m options.
`SQL-script-file`	Names the output file containing the translated information. For use with `transaction-log` only.

Privileges

You must have the BACKUP DATABASE system privilege to run dbtran against a database server. You do not need any privilege to run dbtran against a transaction log file.

Remarks

The dbtran utility takes the information in a transaction log and places it as a set of SQL statements and comments into an output file. The utility can be run in the following ways:

Against a database server

When dbtran is run against a database server, the utility is a standard client application. It connects to the database server using the connection string specified following the -c option, and places output in a file specified with the -n option.

The following command translates log information from the server demo17 and places the output in a file named `demo.sql`.

```
dbtran -c "Server=demo17;DBN=demo;UID=DBA;PWD=sql" -n demo.sql
```

Against a transaction log file

When dbtran is run against a transaction log, the utility acts directly against a transaction log file. Protect your transaction log file from general access to prevent users from having the capability of running this command.

```
dbtran demo.log demo.sql
```

When the dbtran utility runs, it displays the earliest log offset in the transaction log. This can be an effective method for determining the order in which multiple log files were generated.

If -c is used, dbtran attempts to translate the online transaction log file, and all the offline transaction log files in the same directory as the online transaction log file. If the directory contains transaction log files for more than one database, dbtran may give an error. To avoid this problem, ensure that each directory contains transaction log files for only one database.

A transaction can span multiple transaction logs. If transaction log files contain transactions that span logs, translating a single transaction log file (for example `dbtran demo.log`) can cause the spanning transactions to be lost. Use the -m option to generate complete transactions when transactions might span multiple log files. The -m option instructs dbtran to generate a single SQL file (named by -n) containing all the transactions from all the logs in the specified directory.

You can access the Log Translation utility in the following ways:

- From SQL Central, using the *Translate Log File Wizard*.
- At a command prompt, using the dbtran command. This is useful for incorporation into SQL script or command files.

Exit codes are 0 (success) or non-zero (failure).

Related Information

Alphabetical List of Connection Parameters [page 45]
Database Recovery with Multiple Transaction Logs [page 979]
auditing Option [page 688]
Software Component Exit Codes

1.7.4.18 Performance Statistics Utility (dbstats) (UNIX/Linux)

Returns performance statistics for database servers on UNIX and Linux.

Syntax

dbstats [options] [interval] [iterations]

Option	Description
@data	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
-d statistic-definition-string	Specifies the statistic name and the scope (database server, database, or connection) at which the statistic is monitored.
-e server-definition-string	Specifies a comma-separated list of database server names to monitor. If this option is not specified, the dbstats utility reports statistics only for the default database server.
-l	Lists the statistics that are available to be monitored.
-o filename	Writes output messages to the named file.
-v verbosity	Reports messages at the specified level. The supported values are *error*, *warning*, and *info*. The default level is info.
interval	Specifies how often, in seconds, statistics are reported. If you only specify interval, then the utility returns the value of statistics at the specified interval until you stop the reporting of statistics.
iterations	Specifies the total number of times statistics are reported. You must specify an interval value if you specify a value for iterations.

Privileges

None.

Remarks

The dbstats utility uses a shared memory connection to monitor servers running on the same computer. This utility cannot be used to monitor remote database servers.

A `statistic-definition-string` takes the following form:

```
[ [ statistic-name  ][ ,[ statistic-name  ],[ ... ] ] ] [  :[ database-servers ]/
[ databases ]/[ connections ] ]
```

The values `database-servers`, `databases`, and `connections` are comma-separated lists of database server names, database names, and connection names to monitor. In addition to specifying a comma-separated list of names, you can specify an asterisk (*), which indicates that all database servers, databases, and connections should be monitored. If you do not specify a value for one of the supported scopes, then that scope is not monitored.

If no options are specified for the dbstats utility, the utility automatically monitors the statistics ActiveReq, ConnCount, CurrentCacheSize, DiskRead, DiskWrite, MainHeapPages, and UnschReq, for all scopes in which they are valid.

Exit codes are 0 (success) or non-zero (failure).

Example

Query only the ActiveReq statistic for all databases and connections on the default database server and then exit:

```
dbstats -d ActiveReq
```

Query only the ActiveReq statistic for all databases and connections on the default database server, once per second forever:

```
dbstats -d ActiveReq 1
```

Query only the ActiveReq statistic for all databases and connections on the default database server five times, once per second, then exit:

```
dbstats -d ActiveReq 1 5
```

Query the ActiveReq and ConnCount statistics for all databases and connections and then exit:

```
dbstats -d "[ ActiveReq, ConnCount ]"
```

Query the ActiveReq and ConnCount statistics at the database server scope only and then exit:

```
dbstats -d "[ActiveReq, ConnCount]:*"
```

Query the ActiveReq and ConnCount statistics at the database scope only for databases named db1 and db2 running on the default server and then exit:

```
dbstats -d "[ActiveReq, ConnCount]:/db1,db2/"
```

Related Information

Connection, Database, and Database Server Properties [page 879]
Software Component Exit Codes

1.7.4.19 Ping Utility (dbping)

Tests connections to database servers and databases.

Syntax

```
dbping [ options ]
```

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
`-c "keyword=value;..."`	Specifies the connection parameters used to connect to the server and optionally the database. By default, dbping connects to a running server; specify the -d option to connect to the database.

Option	Description
-d	Tests connecting to the database. If necessary, dbping can automatically start the server and database. With -d, dbping behaves like a regular client-application establishing a connection. If you supply the -d option, then dbping reports success if it connects to the server and to the database. If you do not supply the -d option, then dbping reports success if it finds the server. For example, if you have a database server named blair running the sample database, the following succeeds since dbping can connect to the server blair: `dbping -c "Server=blair;DBN=other"` The following command fails, with the message `Ping database failed -- specified database not found` because the database other was not running: `dbping -c "Server=blair;DBN=other" -d`
-en	Specifies that you want dbping to exit with a failed return code when NULL is returned for any of the properties specified. By default, dbping prints NULL when the value for a property specified by -pc, -pd, or -ps is unknown, and exits with a success return code. This option can only be used with -pc, -pd, and -ps.
-l `library`	Specifies the ODBC library to use (without its file extension). This option avoids the use of the ODBC driver manager, and so is useful on UNIX and Linux operating systems. For example, the following command loads the ODBC driver directly: `dbping -m -c "DSN=SQL Anywhere 17 Demo;PWD=sql" -l dbodbc17` On UNIX and Linux, to use a threaded connection library, you must use the threaded version of the Ping utility, dbping_r.
-m	Establishes a connection using ODBC. By default, the utility connects using the Embedded SQL interface.
-o `filename`	Appends output messages to the named file.

Option	Description
-pc `property`,...	Displays the specified connection properties. Supply the properties in a comma-separated list. You must specify enough connection information to establish a database connection if you use this option. For example, the following command displays the fire_triggers option setting, which is available as a connection property. `dbping -c ... -pc fire_triggers`
-pd `property`[@ `db-name`],...	Displays the specified database properties. Supply the properties in a comma-separated list. For example, the following command displays the page size in use by the database: `dbping -c ... -pd PageSize` Optionally, you can specify the name of a database running on the database server you want to obtain the value from. For each property listed, if the database name is not specified by appending @`db-name` to the property, then the database name used for the previous property is used. The following command displays the page size and collation of the database mydb: `dbping -c ... -pd PageSize@mydb,Collation`
-ps `property`,...	Displays the specified database server properties. Supply the properties in a comma-separated list. You must specify enough connection information to establish a database connection if you use this option. For example, the following command displays the number of licensed seats or processors for the database server: `dbping -c ... -ps LicenseCount`
-q	Runs in quiet mode--messages are not displayed.

Option	Description
-s	Returns information about the performance of the network between the computer running dbping and the computer running the database server. Approximate connection speed, latency, and throughput are displayed. The -c option is usually required to specify the connection parameters to connect to a database on the server. You can only use dbping -s for Embedded SQL connections. This option is ignored if -m or -l is also specified. By default, dbping -s loops through the requests for at least one second for each statistic it measures. A maximum of 200 connect and disconnect iterations are performed, regardless of the time they take, to avoid consuming too many resources. On slower networks, it can take several seconds to perform the minimum number of iterations for each statistic. The performance statistics are approximate, and are more accurate when both the client and server computers are fairly idle.
-st `time`	Returns the same information as the -s option, except that the -st option specifies the length of time, in seconds, that dbping loops through the requests for each statistic it measures. This option allows more accurate timing information to be obtained that -s.
-z	Displays the network communication protocols used to attempt connection, and other diagnostic messages. This option is available only when an Embedded SQL connection is being attempted. That is, it cannot be combined with -m or -l.

Privileges

None.

Remarks

The dbping utility is a tool to help debug connection problems. It takes a full or partial connection string and returns a message indicating whether the attempt to locate a server or connect to a database was successful.

The utility can be used for Embedded SQL or ODBC connections. It cannot be used for jConnect (TDS) connections.

Exit codes are 0 (success) or non-zero (failure).

Related Information

List of Connection Properties [page 880]
Software Component Exit Codes
List of Database Server Properties [page 900]
Troubleshooting: How to Test Embedded SQL and Network Connection Performance (dbping) [page 275]
Troubleshooting: How to Test Connection Strings (dbping) [page 274]

1.7.4.20 Profiler Utility (dbprof)

Starts the Profiler so that you can profile a database or review the contents of a previous profiling session.

Syntax

```
dbprof [ options ]
```

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, use the File Hiding utility (dbfhide) to obfuscate the contents of the configuration file.
-c `"keyword=value; ..."`	Specifies connection parameters for the database that you want connect to and start profiling.
-f `filename`	Specifies the Profiler file (`.sqlap`) to open.
-version	Displays the software version number of the Profiler.

Privileges

Any user can open a Profiler file (`.sqlap`); however, you must have the SYS_RUN_PROFILER_ROLE system role to start profiling a database or to connect to a current profiling session.

Remarks

The Profiler is a development and diagnostic tool that logs the activities that occur in your database in real time.

Example

The following command starts the Profiler and connects to the sample database:

```
dbprof -c "UID=DBA;PWD=sql;DBF=%SQLANYSAMP17%\demo.db"
```

The following command starts the Profiler and opens a saved profiling session:

```
dbprof -f myprofilingsession.sqlap
```

Related Information

SQL Anywhere Profiler [page 1483]
Running a Comprehensive Profiling Session (Profiler) [page 1485]
Sending SAP Support a Snapshot of Your Database Server'S Diagnostics (Profiler) [page 1490]
Running a Targeted Profiling Session (Profiler) [page 1486]
Comparing the Results of Different Executions of Stored Procedures, Functions, Events, and Triggers (Profiler) [page 1488]
Criteria for Specifying the SQL Statements to Collect During a Targeted Profiling Session (Profiler) [page 1487]

1.7.4.21 Server Enumeration Utility (dblocate)

Locates database servers on the TCP/IP network.

Syntax

dblocate `[ options ] [ host ]`

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
-d	Lists the server name and address, for each server found, followed by a comma-separated list of databases running on that server. If the list exceeds 160 characters, it is truncated and ends with an ellipsis (...). Databases that are running on SQL Anywhere 9.0.2 and earlier database servers or that were started with the -dh database option are not listed.

Option	Description
-dn `database-name`	Lists the server name and address, for servers running a database with the specified name. If the list exceeds 160 characters, it is truncated and ends with an ellipsis (...). Databases that are running on SQL Anywhere 9.0.2 and earlier database servers or that were started with the -dh database option are not listed.
-dv	Displays the server name and address, for each server found, listing each database running on that server on a separate line. The list is not truncated, so this option can be used to reveal lists that are truncated when the -d option is used. Databases that are running on SQL Anywhere 9.0.2 and earlier database servers or that were started with the -dh database option are not listed.
-n	Lists IP addresses in the output, rather than computer names. This may improve performance since looking up computer names may be slow.
-o `filename`	Appends output messages to the named file.
-p `port-number`	Displays the server name and address only for servers using the specified TCP/IP port number. The TCP/IP port number must be between 1 and 65535.
-q	Runs in quiet mode--messages are not displayed.
-s `name`	Displays the server name and address only for servers with the specified server name. If this option is used, the -ss option should not be used (if both options are used, it is likely that no matching servers will be found).
-ss `substr`	Displays the server name and address only for servers that contain the specified substring anywhere in the server name. If this option is used, the -s option should not be used (if both options are used, it is likely that no matching servers will be found).
-v	Displays the full server name. By default, dblocate truncates database server names that are longer than 40 bytes. Version 9.0.2 and earlier clients, including dblocate, cannot connect to version 10.0.0 and later database servers with names longer than 40 bytes.

Option	Description
`host`	Lists only database servers running on the computer with the specified IP address or hostname. For example, the following command looks for servers on the computer jfrancis: `dblocate jfrancis` The hostname or IP address can be of any format, regardless of whether -n is specified. For example, consider a server is running on myhost.mycompany.com, which has an IP address of 1.2.3.4. To list only servers running on this computer from any computer with the mycompany.com domain, any of `dblocate myhost`, `dblocate myhost.mycompany.com`, or `dblocate 1.2.3.4` can be used.

Privileges

None.

Remarks

The Server Enumeration utility (dblocate) locates any SQL Anywhere database servers running over TCP/IP on the immediate network, and prints a list of the database servers and their addresses. This list includes alternate server names.

Depending on your network, it may take several seconds for dblocate to prints its results.

> **i Note**
>
> If a database server is using a TCP/IP port (ServerPort or PORT) other than 2638 on macOS, dblocate will not find it, even if the -p option is used to specify the TCP/IP port.
>
> Database servers in different subnets can only be located if dbns17 is already in use.

Exit codes are 0 (success) or non-zero (failure).

The database server can register itself with an LDAP server, which keeps track of all servers in an enterprise. This allows both clients and dblocate to find them, regardless of whether they are on a WAN or LAN, through firewalls, and without specifying an IP address. LDAP is only used with TCP/IP, and only on network servers.

If the same database server name is found more than once, dblocate displays the IP address of each host, even if the -n option is not specified. The same server name could be found when a server is running on a computer with multiple IP addresses (for example, if the computer has multiple network cards), or if a network server is running on a remote computer and a personal server with the same name is running on the local computer.

Database servers started with -sb 0 are not found by dblocate.

Related Information

Connections Using LDAP as a Name Server [page 181]
Troubleshooting: How the Broadcast Repeater Utility Locates Database Servers [page 266]
-dh Database Option [page 550]
ServerPort (PORT) Protocol Option [page 235]
Software Component Exit Codes
Troubleshooting: How Database Servers Are Located [page 263]
db_string_ping_server Function

1.7.4.22 Server Licensing Utility (dblic)

Applies your software license to your database server or MobiLink server.

Syntax

```
dblic [ options ] license-file [ "user-name" "company-name" ]
```

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.

Option	Description
-l `type`	Specifies the license type that matches the licensing model described in your software license agreement. The following license types are supported: **Perseat** A perseat license restricts the number of client connections to the database server. With per-seat licensing, the network database server can use all CPUs available on your computer, but not more than the maximum allowed by the SQL Anywhere edition you are running. If a CPU contains multiple cores or threads, all cores and threads of the CPU might be used. The personal server is limited to four cores on one CPU. If a core contains multiple threads, all threads of that core might be used. **Core** A core license restricts the number of separate cores that can be used by the database server. With core-based licensing, the network database server uses up to the number of cores specified in your license, but not more than the maximum allowed by the SQL Anywhere edition you are running. When you have a core license, there are no restrictions on the number of client connections to the database server. You can further restrict the database server with the: • -gt database server option • -gta database server option • -gtc database server option • sa_server_option system procedure with the ProcessorAffinity option This option cannot be used with the -k option.
-k `registration-key`	Specifies a license key. You can use this option to change your database server's edition. For example, if you have the developer edition and want to change to the workgroup edition, you can specify the workgroup edition registration key. This option cannot be used with the -l option.
-o `filename`	Appends output messages to the named file.
-q	Runs in quiet mode--messages are not displayed.

Option	Description
-u `license-number`	Specifies the total number of users or processors for the license. If you are adding extra licenses, this is the total, `not` the number of additional licenses.
`license-file`	Specifies the path and file name of the server executable or license file for the personal database server, network database server, or MobiLink server you are licensing. You can view the current license information for a server executable by entering only the license file name.
`user-name`	Specifies the user name for the license. This name appears in the database server messages window on startup. If there are spaces in the name, enclose it in double quotes.
`company-name`	Specifies the company name for the license. This name appears in the database server messages window on startup. If there are spaces in the name, enclose it in double quotes.

Privileges

None.

Remarks

The Server Licensing utility adds licensed users or licensed processors to your SQL Anywhere database server or MobiLink server. You must use this utility only in accordance with your license agreement to license the number of users or processors to which you are entitled. Running this command does not grant you license. The number of CPUs that the database server can use may also be affected by your SQL Anywhere edition, the -gt, -gtc, or -gta database server options, or the sa_server_option system procedure with the ProcessorAffinity option.

This utility also modifies the user and company names displayed at startup by the personal or network database servers, and the MobiLink server.

You can also use this utility to view the current license information for a personal or network database server by entering only the license file name.

Licensing information is stored in a `.lic` file in the same directory as the server executable. The server looks for a `.lic` file that has the same base file name as the executable that is being run. For example, if the

database server executable was named `myserver.exe`, then the server looks for a license file named `myserver.lic`. By default, the following names are used:

Executable	License file name
Personal database server (dbeng17)	*dbeng17*`.lic`
Network database server (dbsrv17)	*dbsrv17*`.lic`
MobiLink server (mlsrv17)	*mlsrv17*`.lic`

When you attempt to start a server, if the corresponding `.lic` file is not available, then the server does not start. The license file is created by the SQL Anywhere installation program. The dblic utility only modifies existing licenses; it does not create new license files.

Exit codes are 0 (success) or non-zero (failure).

On UNIX and Linux, the database server executable is not writable by default, so using the Server Licensing (dblic) utility fails. Make sure the executable is writable (for example, using `chmod +w`) before you use the Server Licensing utility.

Cached connections count toward per seat licensing.

Example

The following command, run in the same directory as the database server executable, applies a license for 50 users, in the name of Sys Admin, for company My Co, to a Microsoft Windows network database server. The command must be entered all on one line:

```
dblic -l perseat -u 50 "C:\Program Files\SQL Anywhere 17\Bin32\dbsrv17.lic" "Sys
Admin" "My Co"
```

The following messages appear on the screen to indicate the success of the license:

```
SQL Anywhere Server Licensing Utility Version 17.0.11.1293
License applied successfully.
Advanced Edition
Add-on: High Availability
Add-on: In-Memory mode
Add-on: Scale-out Nodes
Licensed nodes: 50
User: Sys Admin
Company: My Co
Install key:
```

The following command returns information about the license for a database server:

```
dblic "C:\Program Files\SQL Anywhere 17\Bin32\dbsrv17.lic"
```

Related Information

Editions and Licensing

-gt Database Server Option [page 446]
-gtc Database Server Option [page 449]
-gta Database Server Option [page 447]
sa_server_option System Procedure
Software Component Exit Codes

1.7.4.23 Service Utility (dbsvc) for Linux

Creates, modifies, and deletes SQL Anywhere services.

Syntax

```
dbsvc [ modifier-options ] -d svc
dbsvc [ modifier-options ] -g svc
dbsvc [ modifier-options ] -l
dbsvc [ modifier-options ] -li
dbsvc [ modifier-options ] -lt
dbsvc [ modifier-options ] -status svc
dbsvc [ modifier-options ] -u svc
dbsvc [ modifier-options ]  creation-options -w svc [ options ]
dbsvc [ modifier-options ] -x svc
```

Major option	Description
-d `service-name`	Removes the named service from the list of services. If you supply -y, the service is deleted without confirmation. You cannot delete a service while it is running.
-g `service-name`	Lists the definition of the service.
-l	Lists the available SQL Anywhere services.
-li	Lists all service interfaces.
-lt	Lists all SQL Anywhere service types.
-u `service-name`	Starts a service named `service-name`.

Major option	Description
-w `executable parameters`	Creates a new service, or overwrites one if one of the same name exists. If you supply -y, the existing service is overwritten without confirmation. You cannot delete a service while it is running. You must supply parameters appropriate for the service you are creating.
-x `service-name`	Stops a service named `service-name`.

Creation option	Description
-a `acct`	Names the account. All services run under a Linux account. If you run under an account you have created, you must name the account with the -a option.
-as	Runs the service under the Linux daemon account. All services run under a Linux account. No password is required. One of -a or -as must be used.
-od	Specifies the location of the system information file (if required).
-pr	Sets the nice level for the Linux process.
-rl	Specifies the runlevels on which to start the service.
-rs `service-name`	Adds the service to the list of services in the Required-Start header of the init script generated by dbsvc. This header helps the operating system determine the order in which it starts services. Specified service names are verified to see if they already exist. The naming convention for the service is SA_`service-name`.
-s `startup`	Sets the startup behavior for SQL Anywhere services. You can set startup behavior to Automatic or Manual. The default is Manual.
-status	Returns the state the service is running.

Creation option	Description
-t `type`	Specifies the type for this service. You can choose from the following types (the alternate name is listed in parentheses): **Network** Creates a service for the SQL Anywhere network database server (dbsrv17). **Personal (Standalone)** This is the default, and creates a service for the SQL Anywhere personal database server (dbeng17). **DBRemote** Creates a service for the SQL Remote Message Agent. **MobiLink** Creates a service for the MobiLink server (mlsrv17). **DBMLSync (MLSync)** Creates a service for the MobiLink synchronization client (dbmlsync). **DBNS** Creates a service for the Broadcast Repeater utility (dbns17). **RSHOST** Creates a service for the Relay Server State Manager. **RSOE** Creates a service for the Outbound Enabler.

Modifier option	Description
-cm	Displays the command used to create the service. This option can be used to output the creation command to a file, which can then be used to add the service on another computer or restore a service to its original state if changes have been made to it. You must specify the -g option or -l option with -cm or the command fails. Specifying -g displays the creation command for the specified service, while specifying -l displays the creation command for all services.
-q	Suppresses messages to the database server messages window. If you specify this option when modifying or deleting an existing service, you must also specify -y or the operation fails.
-y	Automatically performs the action without prompting for confirmation. This option can be used with the -w or -d options. If you specify -q when modifying or deleting an existing service, you must also specify -y or the operation will fail.
-i `interface`	Specify the service interface to use. Value can be one of *systemd* or *lsb*. The default is *systemd* where available; otherwise, the default is *lsb*.

Privileges

None

Remarks

A service runs a database server or other application with a set of options. This utility provides a comprehensive way of managing SQL Anywhere services on Linux.

Because services typically run in a different environment, you must fully qualify the name of the database file when creating a service. Also, do not use spaces in data source names. Spaces and special characters are supported in service names, service arguments, and the SQL Anywhere installation path.

> **i Note**
>
> If your dbsvc command contains characters that are special to the shell (such as semicolons or parentheses), use a configuration file for the parameters to your service.

You must have permissions on the `/etc/init.d` directory to create, edit, or delete services.

When a SQL Anywhere service is running on Linux, a PID file is created in the `/var/run` directory. This file contains the PID of the dbsvc process. The file is named `SA_service-name.pid`. This file can be used by other Linux tools to find the process and monitor the service.

Like most Linux services, the dbsvc utility creates service files in `/etc/init.d`. The naming convention for the service is SA_`service-name`. For example, if you created a service named myserv, you could issue the following command to start the service:

```
/etc/init.d/SA_myserv start
```

The following command returns the status of the service:

```
/etc/init.d/SA_myserv status
```

The following command returns usage information for the service:

```
/etc/init.d/SA_myserv
```

Example

Create a personal server service called myserv, which starts the specified server with the specified parameters. The server runs as the LocalSystem user:

```
dbsvc -as -w myserv -n myeng -c 8m "/tmp/demo.db"
```

Create a network server service called mynetworkserv. The server runs under the local account, and starts automatically when the computer is restarted:

```
dbsvc -as -t network -w mynetworkserv -x tcpip -c 8m "/tmp/demo.db"
```

List all details about service myserv:

```
dbsvc -g myserv
```

Delete the service called myserv, without prompting for confirmation:

```
dbsvc -y -d myserv
```

Create a service called mysyncservice:

```
dbsvc -as -t dbmlsync -w mysyncservice -c "/tmp/CustDB.db" -o syncinfo.txt
```

Generate the command that was used to create the mysyncservice service and output it to the console:

```
dbsvc -cm -g mysyncservice
```

Start a service using dbsvc:

```
dbsvc -u myserv
```

Stop a service using dbsvc:

```
dbsvc -x myserv
```

Obtain the status of a service using dbsvc:

```
dbsvc -status myserv
```

Related Information

Configuration Files [page 1117]
SQL Anywhere Database Server Executable (dbsrv17, dbeng17) [page 380]
SQL Remote Message Agent Utility (dbremote)
mlsrv17 Syntax
MobiLink SQL Anywhere Client Utility (dbmlsync) Syntax
Broadcast Repeater Utility (dbns17) [page 1128]
Relay Server State Manager (rshost) Command-line Syntax (Linux)
Relay Server Outbound Enabler Syntax

1.7.4.24 Service Utility (dbsvc) for Windows

Creates, modifies, and deletes SQL Anywhere services.

Syntax

```
dbsvc [ modifier-options ] -d svc
```

```
dbsvc [ modifier-options ] -g svc
```

```
dbsvc [ modifier-options ] -l
```

```
dbsvc [ modifier-options ] -u svc
```

```
dbsvc [ modifier-options ]  creation-options -w svc details
```

```
dbsvc [ modifier-options ] -x svc
```

```
details:
```

```
<full-executable-path> [ options ]
```

Major option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
-d `service-name`	Removes the named service from the list of services. If you supply -y, the service is deleted without confirmation. You cannot delete a service while it is running.
-g `service-name`	Lists the definition of the service, not including the password.
-l	Lists the available SQL Anywhere services.
-u `service-name`	Starts the service named `service-name`.
-w `executable parameters`	Creates a new service, or overwrites one if one of the same name exists. If you supply -y, the existing service is overwritten without confirmation. You cannot delete a service while it is running. You must supply the full path to the executable that you want to use as a service, as the account under which the service is running may not have the appropriate SQL Anywhere installation directory in its path. You must supply parameters appropriate for the service you are creating.
-x `service-name`	Stops the service named `service-name`.

Creation option	Description
-a `acct`	Names the Microsoft Windows account. All services run under a Microsoft Windows account. If you run under an account you have created, you must name the account with the -a option and supply a password with the -p option. The Login as a Service permission is required for all accounts other than the LocalSystem account. If an account does not have the Login as a Service permission enabled, you are prompted to enable it. If the -y option is also specified, dbsvc attempts to grant the Login as a Service permission without prompting you. If the -q option is specified without the -y option, you are not prompted to grant the Login as a Service permission and dbsvc fails.
-as	Runs the service under the Microsoft Windows LocalSystem account. No password is required. One of -a or -as must be used. All services run under a Microsoft Windows account.
-i	Displays an icon that you can double-click to display the database server messages window. **i Note** As of Windows Vista, this is no longer available.
-p `password`	Specifies the password for the account under which the service runs. Use this option with the -a option.
-rg `dependency,...`	Specifies one or more load ordering groups that must be started before the service being created is allowed to start.
-rs `dependency,...`	Specifies that all the services in the list must have started before the service being created is allowed to start. You can specify the display name or the service name. Service names are verified to see if they already exist. If the specified service name cannot be found, the Service utility (dbsvc) checks to see if there is a service with a matching display name. Create a dependency on the Transport Data Interface (TDI) for TCP/IP to ensure that the connection starts correctly.
-s `startup`	Sets startup behavior for SQL Anywhere services. You can set startup behavior to Automatic, Manual, or Disabled. The default is Manual.
-sd `description`	Provides a description of the service. The description appears in the Windows Service Manager.

Creation option	Description
-sn `name`	Provides a name for the service. This name appears in the Windows Service Manager. If you do not specify the -sn option, the default service name is SQL Anywhere - `svc`. For example, the following service is named SQL Anywhere - myserv by default. `dbsvc -as -w myserv "C:\Program Files\SQL Anywhere 17\Bin32\dbeng17.exe"` To have the service name myserv appear in the Windows Service Manager, you need to run the following (entered all on one line): `dbsvc -as -sn myserv -w myserv "C:\Program Files\SQL Anywhere 17\Bin32\dbeng17.exe"`
-t `type`	Specifies the type for this service. You can choose from the following types (the alternate name is listed in parentheses): **Network** Creates a service for the SQL Anywhere network database server (dbsrv17). **Personal (Standalone)** This is the default, and creates a service for the SQL Anywhere personal database server (dbeng17). **DBRemote** Creates a service for the SQL Remote Message Agent. **MobiLink** Creates a service for the MobiLink server (mlsrv17). **DBMLSync (MLSync)** Creates a service for the MobiLink synchronization client (dbmlsync). **DBNS** Creates a service for the Broadcast Repeater utility (dbns17). **DBLSN (LSN)** Creates a service for the MobiLink Listener utility. **DBVSS (VSS)** Creates a service for the SQL Anywhere VSS writer. **RSOE** Creates a service for the Outbound Enabler. **MLAGENT** Creates a service for the MobiLink Agent.

Modifier option	Description
-cm	Displays the command used to create the service. This option can be used to output the creation command to a file, which can then be used to add the service on another computer or restore a service to its original state if changes have been made to it. You must specify the -g option or -l option with -cm or the command fails. Specifying -g displays the creation command for the specified service, while specifying -l displays the creation command for all services. If the specified service does not exist, the command to delete the service is generated. For example, if service_1 does not exist on the computer, `dbsvc -cm -g service_1` would return the following command to delete the service_1 service: `dbsvc -y -d "service_1"` If the service does not use the LocalSystem account, there is no way to retrieve the password, so it is not included in the command that is generated. If you created the service with `-a user -p password`, only `-a user` is included in the output.
-o `filename`	Writes output from the Service utility (dbsvc) to the specified file. The -o option must be specified before the -d, -g, -l, -u, -w, and -x options. For example: `dbsvc -o out1.txt -y -as -w mysvc "%SQLANY17% \bin32\dbsrv17" -n mysrv -o c:\out2.txt` In this case, the output from dbsvc is logged to `out1.txt`, and the output from the database server is logged to `c:\out2.txt` when the database server is started.
-q	Suppresses messages to the database server messages window. If you specify -q, it is also recommended that you specify a file where messages are logged using the -o option. If you specify this option when modifying or deleting an existing service, you must also specify -y or the operation will fail.
-y	Automatically performs the action without prompting for confirmation. This option can be used with the -w or -d options. If you specify -q when modifying or deleting an existing service, you must also specify -y or the operation will fail.

Remarks

A service runs a database server or other application with a set of options. This utility provides a comprehensive way of managing SQL Anywhere services on Windows. You must be a member of the Administrators group on the local computer to use the Service utility (dbsvc).

You can access the Service utility in the following ways:

- From SQL Central, using the *Create Service Wizard*.
- At a command prompt, using the Service utility (dbsvc).

Exit codes are 0 (success) or non-zero (failure).

Example

Create a personal server service called myserv, which starts the specified server with the specified parameters. The server runs as the LocalSystem user:

```
dbsvc -as -w myserv
"C:\Program Files\SQL Anywhere 17\Bin32\dbeng17.exe"
-n myeng -c 8m "c:\temp\mysample.db"
```

Create a network server service called mynetworkserv. The server runs under the local account, and starts automatically when the computer is restarted:

```
dbsvc -as -s auto -t network -w mynetworkserv
"C:\Program Files\SQL Anywhere 17\Bin32\dbsrv17.exe"
-x tcpip -c 8m "c:\temp\mysample.db"
```

List all details about service myserv:

```
dbsvc -g myserv
```

Delete the service called myserv, without prompting for confirmation:

```
dbsvc -y -d myserv
```

Create a network database server service that uses TLS encryption and also accepts local shared memory connections. The service is started manually.

```
dbsvc -as -s manual -sd "Mail system DB service" -sn "Mailer Database" -t
network -w MailDBService
"C:\Program Files\SQL Anywhere 17\bin64\dbsrv17.exe"
-n mail
-x tcpip
-ec "tls(fips=yes;identity=C:\Program Files\SQL Anywhere 17\samples\certificates
\rsaserver.id;identity_password=test)"
-es c:\mailsystem\mail.db
```

Generate the command to recreate the MailDBService service and output it to a file called `restoreservice.bat`:

```
dbsvc -cm -g MailDBService > restoreservice.bat
```

Create a service dependent on the Workstation service and the TDI group:

```
dbsvc -rs lanmanworkstation -rg TDI -w ...
```

Create two services, with one dependent on the other:

```
dbsvc -sn Aardvark1 -as -w aardsvc1 dbeng17 c:\databases\aard1.db
dbsvc -sn Aardvark2 -as -rs aardsvc1 -w aardsvc2 dbeng17 c:\databases\aard2.db
```

When the Aardvark2 service is started (aardsvc2), the Aardvark1 service (aardsvc1) will automatically start since Aardvark2 is dependent on Aardvark1. Similarly, if the Aardvark1 service must be stopped (aardsvc1), then the Aardvark2 service (aardsvc2) must be stopped first.

Create a service called mysyncservice:

```
dbsvc -as -s manual -t dbmlsync -w mysyncservice
```

```
"C:\Program Files\SQL Anywhere 17\Bin32\dbmlsync.exe"
-c "SQL Anywhere 17 CustDB;PWD=sql"
```

Create a MobiLink listener service that is started manually:

```
dbsvc -as -i -w myListener
"C:\Program Files\SQL Anywhere 17\Bin32\dblsn.exe" "@c:\temp\dblsn.opt"
```

Start the myListener service:

```
dbsvc -u myListener
```

Stop the myListener service:

```
dbsvc -x myListener
```

Create a Volume Shadow Copy Service (VSS) service that is started automatically when the database server starts:

```
dbsvc -as -s Automatic -t vss -w SAVSSWriter
"C:\Program Files\SQL Anywhere 17\Bin32\dbvss17.exe"
```

Related Information

SQL Anywhere Volume Shadow Copy Service (VSS) [page 945]
Windows Services [page 355]
Creating Windows Services (SQL Central) [page 356]
SQL Anywhere Database Server Executable (dbsrv17, dbeng17) [page 380]
SQL Remote Message Agent Utility (dbremote)
mlsrv17 Syntax
MobiLink SQL Anywhere Client Utility (dbmlsync) Syntax
Broadcast Repeater Utility (dbns17) [page 1128]
Relay Server State Manager (rshost) Command-line Syntax (Linux)
Relay Server Outbound Enabler Syntax
Software Component Exit Codes

1.7.4.25 Start Server in Background Utility (dbspawn)

Starts a database server in the background.

Syntax

```
dbspawn [ options ] server-command
```

Option	Description
-f	Forces dbspawn to start a database server, even if a default database server already exists.
-p	Specifies that the operating system process ID of the database server process be returned after the database server starts. For example: `New process ID is 306`
-q	Runs in quiet mode--messages are not displayed.
`server-command`	Specifies the command line for starting the database server.

Privileges

None.

Remarks

The dbspawn utility is provided to start a server in the background. This utility starts the server in the background and returns with an exit code of 0 (success) or non-zero (failure). If a new server is successfully started, dbspawn does not return until the database server has completed initialization and is ready to accept requests.

By default, dbspawn cannot start a new server on a computer where a default database server is currently running. To start a server on a computer where a default database server is running, specify the -f option.

The dbspawn utility is useful for starting a server from a batch file, especially when subsequent commands in the batch file require a server that is accepting requests.

If the specified path includes at least one space, you must enclose the path in one set of double quotes. For example:

```
dbspawn dbeng17 "c:\my databases\mysalesdata.db"
```

If the specified path does not contain spaces, then quotes are not required.

The dbspawn utility does not support the `@data` option itself but you can specify the option in the command to be spawned. The following is an example.

```
dbspawn dbeng17 @myconfig.ini
```

Related Information

SQL Anywhere Database Server Executable (dbsrv17, dbeng17) [page 380]
Software Component Exit Codes

1.7.4.26 Stop Server Utility (dbstop)

Stops a database or database server.

Syntax

dbstop `[ options ] [ server-name ]`

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
-c `"keyword=value;..."`	Specifies a connection string. When stopping a network server, the connection string must include a user ID that has permission to stop the server. By default, SERVER OPERATOR system privilege is required to stop a network server, and all users can shut down a personal server, but the -gk server option can be used to change this. If you supply connection parameters, do not supply a server name as well.
-d	Does not stop the database server. Instead, only stop the database specified in the connection string. The -gd server option controls the permissions required to stop a database file.
-o `filename`	Appends output messages to the named file.
-q	Runs in quiet mode--messages are not displayed.
-x	Does not stop the server if there are still active connections to the server. Including this option prevents dbstop from prompting for confirmation if there are active connections.
-y	Stops the server even if there are still active connections to the server. This is equivalent to including Unconditional=YES in the connection parameters.

Option	Description
`server-name`	Specifies the name of a database server running on the current computer. The database server must be started so that no permissions are required to shut it down. The personal database server starts in this mode by default. For the network database server, you must supply the *-gk all* option. If you supply a server name, do not supply connection parameters as well.

Privileges

When stopping a network server, you must supply a connection string with a user ID that has permissions to stop the server. By default, the SERVER OPERATOR system privilege is required to stop the network server. All users can shut down a personal server. You can use the -gk server command-line option to change the default behavior.

Remarks

The Stop Server utility stops a database server. You can use the -d option to stop a specified database.

The Stop Server utility can only be run at a command prompt. In windowed environments, you can stop a database server by clicking *Shut Down* on the database server messages window.

Options let you control whether a server is stopped, even if there are active connections, and whether to stop a server or only a database.

The behavior of dbstop can be controlled if there are active connections on a server. If there are active connections, dbstop provides a prompt to shut down the server. The -x and -y options can be used to change this behavior.

If dbstop is able to stop the database server, dbstop does not complete until all databases have stopped running, and the database server has been stopped enough so that another server could be started with the same name and databases. When dbstop successfully completes, the database server process may still be running, and some of its resources, such as the output file specified by the -o server option, may still be in use.

Exit codes are 0 (success) or non-zero (failure).

To use the SQLCONNECT environment variable with dbstop, specify the -c option. Otherwise, you can get unexpected results.

Example

You are running the server named myserver without a database. To stop the server, specify the utility database name and the user ID and password that was specified by the -su database server option when the server was started (for example, `-su udba,upasswd`):

```
dbstop -c "Server=myserver;DBN=utility_db;UID=udba;PWD=upasswd"
```

You are running the server named myserver with the database *demo.db* started. To stop the server and database:

```
dbstop -c "UID=DBA;PWD=sql;Server=myserver"
```

You are running a personal server named myserver. To stop the server and databases even if there are connections:

```
dbstop -y myserver
```

You are running a server named myserver with the database *demo.db*. To stop only the database named demo, but not other databases or the server itself, run the following command:

```
dbstop -c "UID=DBA;PWD=sql;Server=myserver;DBN=demo" -d
```

Related Information

Configuration Files [page 1117]
Alphabetical List of Connection Parameters [page 45]
Software Component Exit Codes
STOP SERVER Statement
File Hiding Utility (dbfhide) [page 1160]
Unconditional (UNC) Connection Parameter [page 114]
-gd Database Server Option [page 430]
-gk Database Server Option [page 433]

1.7.4.27 Support Utility (dbsupport)

Sends performance data about errors and diagnostic information to Technical Support.

Syntax

dbsupport `[ options ] operation [ operation-specific-option ]`

dbsupport `configuration-options`

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
`-o filename`	Sends log output messages to the specified file.
-q	Displays only critical log output messages.

Operation	Description
-e `configuration-option`	Displays the setting for the specified configuration option. For example, suppose you ran the following command to configure dbsupport to prompt when possible: `dbsupport -cc promptdefy` When you run the command `dbsupport -ecc`, the following setting is returned: `-cc "promptdefy"` If the current value of the configuration option has never been explicitly set, nothing is echoed.
-is `submission-ID` [*-rr* `N`]	Checks the status of a crash report that has been submitted to Technical Support. For example, the following command returns the status of submission ID 66: `dbsupport -is 66`

Operation	Description
-iu [*-r* `N`]	Checks for updates to your software build. Updates include: **Support Packages** A Support Package is a subset of the software with one or more bug fixes. The bug fixes are listed in the release notes for the update. Bug fix updates may only be applied to installed software with the same version number. While some testing has been performed on the software, do not distribute these files with your application unless you have thoroughly tested your application with the software. **Minor releases** A minor release is a complete set of software that upgrades installed software from an older version with the same major version number (version number format is `major.minor.build`). Bug fixes and other changes are listed in the release notes for the upgrade. You can also check for updates using Interactive SQL or SQL Central.
-lc	Generates a list of all crash reports that have not been submitted to Technical Support. The report names listed can be used with the -sc option.
-ls	Generates a list of submission IDs for all reports that have been submitted. For example: `dbsupport -ls` This command returns information similar to the following: `SQL Anywhere Support Utility Version 17.0.11.1293` `Submission ID: 217756 reported: Wed Jul 29 15:54:43 2015` `Submission ID: 217757 reported: Wed Jul 29 15:57:07 2015` `Submission ID: 217759 reported: Wed Jul 29 16:14:24 2015` `Submission ID: 217760 reported: Wed Jul 29 16:14:30 2015` `Submission ID: 217761 reported: Wed Jul 29 16:28:12 2015` `Submission ID: 221030 reported: Mon Sep 07 12:34:59 2015` `Submission ID: 221031 reported: Mon Sep 07 12:35:05 2015`

Operation	Description
-pc `filename`	Displays crash report information. You can use this option to view information before it is submitted.
-pd	Displays the diagnostic information that has been collected. You can use this option to view information before it is submitted.
-ps `submission-ID`	Displays information about a specific report that has been submitted. For example, running the command `dbsupport -ps 4` returns information about submission 4: `SQL Anywhere Support Utility Version 17.0.11.1293` `Submission ID: 217756  reported: Wed Jul 39 15:54:43 2015`
-sa [*-r* `number-of-submission-retries`]	Submits all crash report and diagnostic information stored in the diagnostic directory.
-sc `reportname` [*-r* `number-of-submission-retries`] [*-nr* | *-rr* `N`]	Submits a crash report and diagnostic information. For example: `dbsupport -sc SA17_20151220_133828_32116` Use the -lc option to see a list of reports that have not been submitted.
-sd [*-r* `number-of-submission-retries`]	Submits only diagnostic information.

Configuration option	Description
-cc [*autosubmit* | *no* | *promptDefY* | *promptDefN*]	Changes the prompting behavior of dbsupport. This option is You can specify one of the following options: **autosubmit** Submit crash reports automatically and submit diagnostic reports regularly. **no** Do not prompt for permission to submit reports. Reports are not submitted. **promptDefY** If possible, prompt for permission to submit the report. If no answer is given, submit the report. **promptDefN** If possible, prompt for permission to submit the report. If no answer is given, do not submit the report. This is the default. For example, if you are using embedded SQL Anywhere in an application, you may want to configure the Support utility to not submit reports. If you specify the -cc option, its value becomes the default value used by the Support utility (dbsupport). The setting is stored in the `dbsupport.ini` file in the diagnostic directory. The following command configures the Support utility so that it does not submit reports and never prompts the user to submit reports: `dbsupport -cc no`
-cd `retry-delay`	Specifies the retry delay, in seconds, for submitting a report if the previous attempt is unsuccessful. The default delay is 30 seconds. If you specify this option, its value becomes the default value used by the Support utility. The setting is stored in the `dbsupport.ini` file in the diagnostic directory. The following command specifies that failed submissions should be retried every three seconds, up to a maximum of four times before giving up: `dbsupport -cr 4 -cd 3`

Configuration option	Description
-ce `email-address;email-server[:port][;user-id:password]`	Specifies the address where an email is sent after a crash occurs. The email is sent using the `email-server` SMTP server. Optionally, you can specify the port that should be used and the user ID and password used to authenticate with the SMTP server.
-cet	Tests the email settings specified by the -ce option.
-cid `customer-id`	Specifies a string that identifies you in the submission report. If you specify this option, its value becomes the default value used by the Support utility. The configuration is stored in the `dbsupport.ini` file in the diagnostic directory. The following examples specify a customer identification string for dbsupport: `dbsupport -cid myid@company.com` `dbsupport -cid "MyClientApp 1.0"`
-cid-	Removes the customer identification string from the `dbsupport.ini` file.
-cp { `host` [`:port`] | *autodetect* }	Configures the HTTP proxy host and port used to submit error reports. The default port is 80. If you specify `-cp autodetect` on Microsoft Windows and a proxy server and port have been set for Microsoft Internet Explorer, then dbsupport configures its proxy server and port using the system setting. If you specify `-cp autodetect` on UNIX and Linux and the HTTP_PROXY environment variable is set, then dbsupport configures its proxy server and port using the environment variable setting.
-cp-	Removes HTTP proxy host and port settings from the `dbsupport.ini` file.

Configuration option	Description
-cr `number-of-submission-retries`	Specifies the number of times a failed submission should be retried. If you specify this option, its value becomes the default value used by the Support utility. The setting is stored in the `dbsupport.ini` file in the diagnostic directory. The following command specifies that failed operations should be retried every three seconds, up to a maximum of four times before giving up: `dbsupport -cr 4 -cd 3`

Operation-specific option	Description
-ac	Adds a comment to the submission.
-af	Attaches a file to the submission. You can specify the -af option more than once to attach multiple files to the submission.
-nr	Specifies that dbsupport does not check the database server for the status of the submission. For example, the following command submits the report, but does not check for the status of the new submission: `dbsupport -nr -sc SA17_20151220_133828_32116` By default, dbsupport checks whether there is already a fix for the problem being submitted.
-r `number-of-submission-retries`	Specifies the maximum number of times dbsupport should try to send the submission. Specifying 0 means retry indefinitely. The default value is 10. Specifying -r overrides the -cr value stored in the `dbsupport.ini` file, if one exists.
-rd `retry-delay`	Specifies the number of seconds dbsupport waits between attempts to resend the report. The default value is 30. Specifying -rd overrides the -cd value stored in the `dbsupport.ini` file, if one exists.
-rr `number-of-submission-response-retries`	Specifies the maximum number of times dbsupport should try to obtain a submission response. Specifying 0 means retry indefinitely. The default value is 10.

Privileges

None.

Remarks

Performance data, such as crash reports and diagnostic information reports, is used to help improve the product.

If a fatal error or crash occurs in any SQL Anywhere program, an error report is created which contains technical information about the program at the time of the error. When an error report is created, you are prompted to submit this report to Technical Support.

You can also submit this report using the Support utility (dbsupport). Submitting error reports assists with diagnosing the cause of a fatal error or assertion. Once an error report is submitted, you receive a submission ID that you can quote to Technical Support. If you report the problem to Technical Support, sending them this error file can speed up the processing of your issue. Submitting error and diagnostic reports helps to improve the product, even if you choose not to contact Technical Support.

The Support utility (dbsupport) can be used for any of the following tasks:

- submitting diagnostic information and crash reports
- submitting feature statistics
- listing information about submitted and unsubmitted crash reports
- printing information about submitted and unsubmitted crash reports
- inquiring about the status of a submission
- inquiring whether there are software updates available for your build of SQL Anywhere
- configuring dbsupport behavior for when a fatal error (assertion/crash) is detected. For example, dbsupport can be configured to run a specified handler program each time the database server submits an error report. This feature is useful for adding your own custom actions to the error handling process.
- configuring how many times to retry sending a report. For example, when submitting a report, it could be configured to retry the operation again in 30 seconds, up to a maximum of ten times. This feature is useful for handling the case where the service may be temporarily unavailable.

Settings for the Support utility are stored in the `dbsupport.ini` file in the diagnostic directory.

The operation-specific options are useful for overriding default behavior, including those that have been saved in the `dbsupport.ini file`.

The graphical administration tools can be configured to remove the option to submit error reports.

If you choose not to submit a report, it remains in the diagnostic directory on your hard disk. This directory is created automatically by the software programs and is used only for saving reports and for storing settings for the Support utility (dbsupport).

Diagnostic directory location

The location of the diagnostic directory varies depending upon the platform.

The database server checks for the diagnostic directory according to the following order of precedence:

Microsoft Windows

1. The directory specified by the SADIAGDIR environment variable, if it is set.
2. The default diagnostic directory location:
   - On Microsoft Windows 7 and later: `%ALLUSERSPROFILE%\`*SQL Anywhere 17*`\diagnostics`.
3. The current directory.

4. The temporary directory. See SATMP environment variable and TMP, TMPDIR, and TEMP environment variables.

On Microsoft Windows, the diagnostic directory is in a location that is writable by all users of the computer. If a computer has a number of user accounts and you need to keep the error reports isolated from other users, set the SADIAGDIR environment variable for each user to point to a directory that is readable and writable only to that particular user.

UNIX, Linux, and macOS

1. The directory specified by the SADIAGDIR environment variable, if it is set.
2. The default diagnostic directory: `$HOME/.sqlanywhere17/diagnostics`.
3. The current directory.
4. The temporary directory. See SATMP environment variable and TMP, TMPDIR, and TEMP environment variables.

> **i Note**
>
> On UNIX and Linux, writing crash reports to the user's home directory is not recommended when the database or MobiLink server is running as a daemon, or the user is root/nobody.

Error report file names

Error report file names are composed as follows:

- a prefix that identifies the application:

Application prefix	Application
ISQL	Interactive SQL utility
LSN	Listener utility
MLC	MobiLink client
MLS	MobiLink server
ML_MON	MobiLink Profiler
SA	Personal or network database server
SCJ	SQL Central
SR	SQL Remote

- a value indicating the software version
- two fields linked with underscores that provide the timestamp for when the error report was created
- the application identifier
- the extension `.dmp`

For example, `SA17_20150620_133828_32116.dmp` is an error report from a SQL Anywhere version 17 database server from 2015/06/20, at 1:38:28 P.M., from process 32116.

Example

1. You can change the default behavior of dbsupport with the -cc option:
   - The following command configures dbsupport to submit reports automatically without prompting the user:

     ```
     dbsupport -cc autosubmit
     ```
   - The following command configures dbsupport to automatically submit reports and to send a notification email to `myemail@company.com` by using the email server `emailserver.company.com`:

     ```
     dbsupport -cc autosubmit -ce
     "myemail@company.com";"emailserver.company.com:25"
     ```
   - The following command turns off automatic report submission:

     ```
     dbsupport -cc no
     ```
2. You can view the list of error reports with the -lc option. The following command generates a list of all crash reports that have not been submitted:

   ```
   dbsupport -lc
   ```
3. You can manually submit error reports with the -sa, -sc, or -sd option. The following command submits all reports stored in the diagnostic directory:

   ```
   dbsupport -sa
   ```
4. You can include comments and files with the error report by specifying the -ac and -af options, respectively:

   ```
   dbsupport -sc SA17_20150901_113308_3360 -ac "The message.txt file provides
   more
   information about this error report." -af c:\scenario.txt -af c:\message.txt
   ```

Related Information

Regularly Submit Performance Data [page 1022]
Software Updates [page 1113]
Administration Tools Configuration
Sending SAP Support a Snapshot of Your Database Server'S Diagnostics (Profiler) [page 1490]
SADIAGDIR Environment Variable [page 575]

1.7.4.28 Transaction Log Utility (dblog)

Administers the transaction log for a database.

Syntax

```
dblog [ options ] database-file
```

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
-ek `key`	Specifies the encryption key for strongly encrypted databases directly in the command. If you have a strongly encrypted database, you must provide the encryption key to use the database or transaction log in any way. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify the correct key for a strongly encrypted database.
-ep	Specifies that you want to be prompted for the encryption key. This option causes a window to appear, in which you enter the encryption key. It provides an extra measure of security by never allowing the encryption key to be seen in clear text. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify the correct key for a strongly encrypted database.
-ft `timeline`	Resets the timeline to `timeline`, so that the database can take part in replication. This option is used for reloading SQL Remote consolidated databases. The timeline must be specified in the format: **`"GUID timestamp"`** where `timestamp` is a UTC representation of time in the format "YYYY-MM-DD HH:MM:SS.MMMM", which is the same format used in dblog output.
-ir	Resets the SQL Remote log offset that is kept for the delete_old_logs option, allowing transaction logs to be deleted when they are no longer needed. Use this option if you have stopped using SQL Remote on this database, but continue to use MobiLink synchronization.
-is	Resets the MobiLink log offset that is kept for the delete_old_logs option, allowing transaction logs to be deleted when they are no longer needed. Use this option if you have stopped using MobiLink synchronization on this database, but continue to use SQL Remote.

Option	Description
-m `mirror-name`	Specifies the file name for a new transaction log mirror. If the database is not currently using a transaction log mirror, it starts using one. If the database is already using a transaction log mirror, it changes to using the new file as its transaction log mirror.
-n	Stops using a transaction log, and stops using a transaction log mirror. Without a transaction log, the database can no longer participate in data replication or use the transaction log in data recovery. If a SQL Remote or dbmlsync truncation offset exists, the transaction log cannot be removed unless the corresponding ignore option (-ir for SQL Remote, or -is for dbmlsync) is also specified. You cannot stop using a transaction log if the database has auditing turned on (unless you first turn auditing off) and the audit_log option is set to require the transaction log.
-o `filename`	Appends output messages to the named file.
-q	Runs in quiet mode--messages are not displayed.
-r	Maintains a single transaction log file, and stops using a transaction log mirror.
-t `log-name`	Specifies the file name for a new transaction log. If the database is not currently using a transaction log, it starts using one. If the database is already using a transaction log, it changes to using the new file as its transaction log.
-x `n`	Resets the transaction log current relative offset to `n`, so that the database can take part in replication. This option is used for reloading SQL Remote consolidated databases.
-z `n`	Resets the transaction log starting offset to `n`, so that the database can take part in replication. This option is used for reloading SQL Remote consolidated databases.

Privileges

None.

Remarks

The dblog utility allows you to display or change the name of the transaction log or transaction log mirror associated with a database. You can also stop a database from maintaining a transaction log or mirror, or start maintaining a transaction log or mirror.

A transaction log mirror is a duplicate copy of a transaction log, maintained by the database in tandem.

The name of the transaction log is first set when the database is initialized. The Transaction Log utility works with database files. The database server must not be running on that database when the transaction log file name is changed (or an error message appears).

The utility displays additional information about the transaction log, including the following:

- Version number
- The name of the transaction log file
- The name of the transaction log mirror file, if any
- The current relative offset

You can access the Transaction Log utility in the following ways:

- From SQL Central, using the *Change Log File Settings Wizard*.
- From Interactive SQL, using the ALTER DATABASE `dbfile` ALTER LOG statement.
- At a command prompt, using the dblog command.

Exit codes are 0 (success) or non-zero (failure).

Related Information

Configuration Files [page 1117]
Remote Database Extraction to a Script File
Timelines [page 988]
Changing the Location of a Transaction Log (SQL Central) [page 294]
Changing the Location of a Transaction Log (Command Line) [page 295]
File Hiding Utility (dbfhide) [page 1160]
ALTER DATABASE Statement
Software Component Exit Codes

1.7.4.29 Unload Utility (dbunload)

Unloads a database into a specified directory.

Syntax

dbunload `[ options ] [ directory ]`

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, use the File Hiding utility (dbfhide) to encode the contents of the configuration file.

Option	Description
-ac `"keyword=value; ..."`	Connects to an existing database and reloads the data directly into it, combining the operations of unloading a database and reloading the results into an existing database. For example, you could create a new database using the dbinit utility, and then reload it using this option. This method is useful when you want to change initialization options. The following command loads a copy of the `data.db` database into an existing database file named `backup.db`: `dbunload -c "UID=DBA;PWD=passwd;DBF=data.db" -ac "UID=DBA;PWD=passwd;DBF=backup.db"` If the new database must have the same file name as the original database, then use the DBN connection parameter to give each database a unique temporary name. The following command loads a copy of the old database into a new database that has the same file name `data.db`. `dbunload -c "UID=DBA;PWD=passwd;DBN=old;DBF=\master\data.db" -ac "UID=DBA;PWD=passwd;DBN=new;DBF=\backup\data.db"` If the original database was created using version 9 or earlier of SQL Anywhere and the new database is not already running, you must provide a database server name in the -ac option. For example: `dbunload -c "UID=DBA;PWD=passwd;DBF=data.db" -ac "UID=DBA;PWD=passwd;DBF=newdata.db;Server=newserver"` If you specify this option, then no interim copy of the data is created on disk, so do not specify an unload directory in the command. This behavior provides greater security for your data. If you specify this option, then passwords are automatically unloaded; you do not have to specify the -up option.

Option	Description
-ae	Use this option when rebuilding a database (for example, the -ac, -an, -ar options) to specify that the unload operation proceed even if errors are encountered. When -ae is specified, dbunload executes all statements in the `reload.sql` script file. If any statements result in an error, then they are skipped and logged in the `unprocessed.sql` script file. They are also displayed in the command line output (unless the -q option is specified). After attempting to rebuild a database that encountered errors, review and fix the errors and rebuild the database from scratch (recommended), or fix them in the `unprocessed.sql` script file and apply them to the unfinished database. This option is not for use when rebuilding online databases (for example, -ao or -aob options).
-an `database`	Combines the operations of unloading a database, creating a new database, and loading the data using this option. The options specified when you created the source database are used to create the new database. However, you can change the initialization options as necessary by specifying other supported dbunload options (such as -ap to change the page size or -et to enable table encryption). For example, the following command creates a new database file named `copy.db` and copies the schema and data of `data.db` into it: `dbunload -c "UID=DBA;PWD=passwd;DBF=data.db" -an copy.db` If you use this option, no interim copy of the data is created on disk, so you do not specify an unload directory in the command. This provides greater security for your data. When the new database is created, the dbspace file names have an R appended to the file extension to prevent file name conflicts if the dbspace file for the new database is created in the same directory as the dbspace for the original database. For example, if an unloaded database has a dbspace called library in the file `library.db`, then the library dbspace for the new database is `library.dbR`. The file specified by -an is relative to the database server. If you specify this option, then passwords are automatically unloaded; you do not have to specify the -up option.

Option	Description
-ao `database-name`	Performs a full backup and an incremental backup to produce a rebuilt database to which you can apply log files from the production database. To use the -ao option, the following conditions must be met: • The original database must be created by SQL Anywhere version 17 . • The page size, encryption algorithm, and encryption key of the rebuilt database must be the same as the original database. • The computer where the rebuild is run must have enough space to hold twice the total of the database, dbspaces, and the log file of the original database as intermediate files are required. • If any dbspaces are in use by the current database, then the dbspace files must be in the same directory as the database file and must not use an absolute path. • The database must not be using high availability. If the conditions below cannot be met, then consider rebuilding the database using dbunload -aob: • There must be regular times when there are no outstanding transactions on the production server as `dbbackup -wa` must be used to take the initial backup to ensure that no transactions are lost. • Multiple backups to the production server and transaction log renames on the production server must be acceptable. When using the -c option with the -ao option, the connection string is for the production database and is used to back up the initial database before the rebuild and to create an incremental transaction log backup, which is applied to the rebuild. Database encryption settings cannot be changed during an online rebuild. Settings from the original database are automatically preserved. If the database is using strong encryption, then use the -ek or -ep dbunload option to provide the current encryption key. The rebuilt database cannot be restarted unless a copy of the production database transaction log is located in the same directory as the database file. The rebuilt database's transaction log path is in the same directory as the database file, regardless of any relative or absolute path in the original production database.

Option	Description
-aob	Creates a rebuild database to which you can apply transaction logs from the production database. If there are no regular times when there are no outstanding transactions by any user on the production server, or if multiple backups to the production server and transaction log renames on the production server are not acceptable, then perform a full backup with a transaction log rename, ensuring that there are no transactions in progress during the log rename. Then you can do an online rebuild from that backup by using dbunload -aob, where the connection string is to the backup database, rather than the production database. You can then apply the transaction log from the production database after the backup. The rebuilt database's transaction log path is in the same directory as the database file, regardless of any relative or absolute path in the original production database. Database encryption settings cannot be changed during an online rebuild. Settings from the original database are automatically preserved. If the database is using strong encryption, then use the -ek or -ep dbunload option to provide the current encryption key. **i Note** Use the -ao option to rebuild a database whenever possible because it includes an initial full backup, rebuild, and one or more iterations of applying the transaction log. There are fewer steps required both before and after the rebuild when using the -ao option as compared with using the -aob option.
-aot `seconds`	Specifies the maximum time for an incremental backup of the production database and the rebuilt database. This option is ignored unless -ao is also used. If the incremental backup and apply time is greater than the value specified, then another incremental backup and apply is performed. To stop the incremental backup and apply at any time, press Q and dbunload stops after applying the backup. **i Note** Specifying `-aot 0` is not valid. Specifying a value for `seconds` that is less than one minute can cause indefinite looping (until the Q key is pressed), and also can result in hundreds of transaction log renames on the production database.

Option	Description
-ap `size` [*k*]	Sets the page size of the new database. For example, the following command unloads a database, old.db, into a new database, new.db, and changes the page size to 8k: `dbunload -c "UID=DBA;PWD=passwd;DBN=old;DBF=old.db" -an new.db -ap 8k` This option is ignored unless -an or -ar is also used. The page size for a database can be (in bytes) 2048 (deprecated), 4096, 8192, 16384, or 32768, with the default being the page size of the original database. Use *k* to specify units of kilobytes (for example, -ap 4k). You must specify either -an or -ar with this option. If there are already databases running on the database server, the server's page size (set with the -gp option) must be large enough to handle the new page size.

Option	Description
-ar [`directory`]	Creates a new database with the same settings as the old database, reloads it, and replaces the old database. However, you can change the initialization options as necessary by specifying other supported dbunload options (such as -ap to change the page size or -et to enable table encryption). If you use this option, there can be no other connections to the database, and the database connection must be local, not over a network. If you specify an optional `directory`, the transaction log offsets are reset for replication purposes, and the transaction log from the old database is moved to the specified directory. The named directory should be the directory that holds the old transaction logs used by the Message Agent. The transaction log management is handled only if the database is used in replication: if there is no SQL Remote publisher, then the old transaction log is not needed and is deleted instead of being copied to the specified directory. When the new database is created, the dbspace file names have an R appended to the file name to prevent file name conflicts if the dbspace file for the new database is created in the same directory as the dbspace for the original database. For example, if an unloaded database has a dbspace called library in the file `library.db`, then the library dbspace for the new database is `library.dbR`. If you are rebuilding an encrypted database, the encryption key for the original and new databases must be the same. Using the -ar option resets the database truncation points to zero. If you specify this option, then passwords are automatically unloaded; you do not have to specify the -up option.
-bp	Groups CREATE INDEX and LOAD TABLE statements inside BEGIN PARALLEL WORK statements in the `reload.sql` file. Statements inside a BEGIN PARALLEL WORK statement execute in parallel.

Option	Description
-c `"keyword=value; ..."`	Specifies the connection parameters for the source database. For example, the following statement unloads the sample database, connecting as user ID DBA with password sql. The data is unloaded into the `c:\unload` directory. `dbunload -c "DBF=%SQLANYSAMP17% \demo.db;UID=DBA;PWD=sql" c:\unload`
-cm { *sql* | dbinit }	Displays in the command prompt the CREATE DATABASE or dbinit command to create a database that is the same as the one being unloaded. If -an is also specified, the command that is displayed is the command to create the new database. **sql** Displays the CREATE DATABASE statement that is written to the `reload.sql` file. **dbinit** Displays the dbinit utility command. When displaying the CREATE DATABASE statement or dbinit command, asterisks appear for the DBA user ID, DBA password, and encryption key (if there is one). The creation command or statement is not displayed if you unload a database that was created with a version 10 or earlier database server.
-cp	Compresses the table data output files by appending the COMPRESSED keyword to the UNLOAD TABLE statements it executes. This option has no effect when specified with -an or -ar.
-d	Does not generate any of the database definition statements (CREATE TABLE, CREATE INDEX, and so on); `reload.sql` contains statements to reload the data only.

Option	Description
-dc	Forces all computed columns in the database to be recalculated. By default, computed column values are not recalculated. When the -dc option is specified, a new section is added to the `reload.sql` script to recompute computed columns. Statements of the following form are added. `ALTER TABLE "owner"."table-name"` `ALTER "computed-column" SET COMPUTE (compute-expression);` If your tables contain context-sensitive computed values, such as CURRENT DATE, use the ALTER TABLE statement to recalculate computed column values instead of using the -dc option.
-dt `directory`	Specifies the temporary directory used during the rebuild. The directory must exist, is relative to the client, and must have enough space available for the production database and transaction log file. There also must be enough space for the newly created database if the temporary directory and the rebuilt database are in the same disk partition. It also specifies the temporary directory for the `unprocessed.sql` file, if it cannot be written to its file path. If this option is not specified, the location for the temporary directory is specified by one of the SATMP, TMP, TMPDIR, or TEMP environment variables.
-e `table`, ...	Excludes the specified tables from the `reload.sql` file. Table names are always case insensitive, even in case sensitive databases. A `reload.sql` file created with the -e option should not be used to rebuild a database because the file will not include all the database tables. If a table has foreign keys referring to it, the database cannot be rebuilt without the contents of the table. Only use the -e option with the -d option to unload data for all tables except those identified by -e. Since the unload file generated does not include database schema users, views, and procedures, the file cannot be used to rebuild a database.

Option	Description
-ea `algorithm`	Specifies the encoding algorithm used for database or table encryption (-et). Specify `-ea simple` for simple obfuscation (do not specify -ek or -ep). Simple obfuscation is intended only to keep data hidden in the event of casual direct access of the database file, to make it more difficult for someone to decipher the data in your database using a disk utility to look at the file. For greater security, specify AES or AES256 for 128-bit or 256-bit strong encryption, respectively. Specify AES_FIPS or AES256_FIPS for 128-bit or 256-bit FIPS-certified encryption, respectively. For strong encryption, you must also specify the -ek or -ep option. To create a database that is not encrypted, specify `-ea none`, or do not include the -ea option (and do not specify -e, -et, -ep, or -ek). If you do not specify the -ea option, the default behavior is as follows: • -ea NONE, if -ek, -ep, or -et is not specified • -ea AES, if -ek or -ep is specified (with or without -et) • -ea SIMPLE, if -et is used without -ek or -ep Algorithm names are case insensitive.
-ek `key`	Specifies an encryption key in the dbunload command for the new database created if you unload and reload a database (using the -an option). If you create a strongly encrypted database, you must provide the encryption key to use the database or transaction log in any way. The algorithm used to encrypt the database is the algorithm specified by the -ea option. If you specify the -ek option without specifying -ea, the AES algorithm is used. Protect your key. Store a copy of your key in a safe location. A lost key will result in a completely inaccessible database, from which there is no recovery.
-ep	Prompts for an encryption key for the new database created if you unload and reload your database using the -an option. It provides an extra measure of security by never allowing the encryption key to be seen in clear text. If you specify -ep without specifying -an, the -ep option is ignored. If you specify -ep and -an, you must input the encryption key twice to confirm that it was entered correctly. If the keys don't match, the unload fails.

Option	Description
-er	Removes encryption from encrypted tables during an unload procedure. When rebuilding a database that has table encryption enabled, you must specify either -er or -et to indicate whether the new database has table encryption enabled, otherwise you get an error when attempting to load the data into the new database. The following command unloads a database (`data.db`) that has encrypted tables, into a new database (`copy.db`) that does not have encryption enabled, removing encryption from any encrypted tables: `dbunload -an copy.db -er -c "UID=DBA;PWD=passwd;DBF=data.db;DBKEY=29bN8cj1z"`
-et	Enables database table encryption in the new database (-an or -ar must also be specified). If you specify the -et option without the -ea option, the AES algorithm is used. If you specify the -et option, you must also specify -ep or -ek. You can change the table encryption settings for the new database to be different than those of the database you are unloading. When rebuilding a database that has table encryption enabled, you must specify either -er or -et to indicate whether the new database has table encryption enabled, otherwise you get an error when attempting to load the data into the new database. The following example unloads a database (`data.db`) that has encrypted tables, into a new database (`copy.db`) that has table encryption enabled, and uses AES_FIPS encryption with the key 34jh: `dbunload -an copy.db -et -ea AES_FIPS -ek 34jh -c "UID=DBA;PWD=passwd;DBF=data.db"`

Option	Description
-g	Controls whether materialized views and indexes are refreshed after a reload.. **Materialized views** By default, materialized views defined as MANUAL REFRESH are not initialized after a reload. If you want these materialized views to be initialized as part of the reload process, specify the -g option. Specifying -g causes the database server to call the sa_refresh_materialized_views system procedure. When deciding whether to use the -g option, consider that initializing all materialized views may cause the reload process to take significantly longer to complete. However, not using the -g option means that the first query that attempts to use an uninitialized materialized view must wait while the database server initializes the view, which may cause an unexpected delay. If you do not use the -g option, you can also manually initialize materialized views after the reload completes. **Text indexes** By default, text indexes defined as MANUAL REFRESH are not initialized after a reload. If you want the text indexes initialized as part of the reload process, specify the -g option. Specifying -g causes the database server to call the sa_refresh_text_indexes system procedure.
-ii	Uses the UNLOAD statement to extract data from the database, and uses the LOAD statement in the `reload.sql` file to repopulate the database with data. This is the default.
-ix	Uses the UNLOAD statement to extract data from the database, and uses the Interactive SQL INPUT statement in the `reload.sql` file to repopulate the database with data.

Option	Description
-k	Populates the sa_diagnostic_auxiliary_catalog table. This table maps database object IDs for tables, users, procedures, and so on, from the source database to the tracing database. It also causes all histograms to be unloaded/reloaded. This option is used when creating a tracing database, that is, a database that receives diagnostic tracing information. Additionally, this option ensures that auditing is disabled on the tracing database, even if auditing is enabled on the production database. The sa_diagnostic_auxiliary_catalog table allows the database server to simulate conditions that were present when tracing data was captured (for example, for use with Index Consultant). This option is most useful when specified with the -n option.
-kd	Reloads the database into a single dbspace file. This option is useful for creating a tracing database if the computer where the tracing database is being created does not have the same directory structure as the production database.
-kdi `number`	Specifies the number of times that the encryption key is hashed. Specify a whole number between 1 and 1000. The default value is 2, which results in the hash being applied 2000 times. The higher the number, the better security. If you are running the database on a slow computer, then a very high number of iterations could result in the database taking longer to start (once the database is running, there is no performance impact). This option must be specified with either the -ek or the ep option, as well as either the -ar or the -an option.
-l	Forces the current value of SYSTABCOL.max_identity to be preserved across a database rebuild. By default, when a database containing tables with AUTOINCREMENT columns is rebuilt, the database server calculates the next available value for each AUTOINCREMENT column based on the current contents of the tables. Usually this is enough; however, if rows have been deleted from the end of the range of values, values can be reused, which is not desirable. Specifying the -l option adds calls to the sa_reset_identity system procedure to the generated `reload.sql` script for each table that contains an AUTOINCREMENT value, preserving the current value of SYSTABCOL.max_identity.
-m	Does not preserve user IDs for databases involved in replication.

Option	Description
-n	Does not unload database data; `reload.sql` contains SQL statements to build the structure of the database only. If you want the `reload.sql` file to contain LOAD TABLE or INPUT statements, use -nl instead.
-nl	Unloads the structure (the same behavior as the -n option), but the resulting `reload.sql` file also includes LOAD TABLE or INPUT statements for each table. No user data is unloaded when this option is used. When you specify -nl, you must also include a data directory so that the LOAD/INPUT statements can be generated, even though no files are written to the directory. This option allows you to generate a reload script without unloading data. You can unload the data by specifying -d. If a database contains a table whose data should not be unloaded, unloading the data for that table can be skipped using `dbunload -d -e table-name`.

Option	Description
-no	Unloads the database objects ordered by name. By default, dbunload generates objects in the order they were created. Specifying the -no option may be useful for comparing database schemas when the databases contain the same objects, but the creation order was different. Object definitions are grouped by object type in alphabetical order in the `reload.sql` file if -no is specified: • users • group memberships • tables • indexes and foreign keys • views • procedures • functions • triggers • events • web services The object definitions are output in owner,name order. Sometimes a third element such as a foreign key, role name, or trigger name is included in the ordering. The -no option cannot be used with the -n, -nl, -ar, -an, or -ac option. To simplify comparisons, use -no option when comparing the reload scripts for databases that were created using the same version of the database server because of minor differences in the object definitions. **⚠ Caution** The generated file should not be used to create a new database because the object creation order can be important. For example, if procedure p2 calls procedure p1 and p1 returns a result set, it may be important to define p1 before p2. Attempting to run a `reload.sql` script file generated with -no option results in an error. An error is returned if this option is specified with the -up option.
-o `filename`	Appends output messages to the named file. The location of this file is relative to the directory in which you run dbunload.
-p `char`	Replaces the default escape character (\) for external unloads (dbunload -x option) with another character. This option is available only when you run this utility at a command prompt.

Option	Description
-q	Runs in quiet mode; messages are not displayed. This option is available only when you run this utility at a command prompt. If you specify -q, you must also specify -y or the unload will fail if `reload.sql` already exists.
-qr	Prevents progress messages from being created and displayed when loading tables and creating indexes.
-r `reload-file`	Modifies the name and directory of the generated reload SQL script file. The default is `reload.sql` in the current directory. The directory is relative to the current directory of the client application, not the database server.
-ru `unprocessed-file`	Modifies the name and directory of the generated unprocessed SQL statements file. The default is `unprocessed.sql` in the current directory. The directory is relative to the current directory of the client application, not the database server.
-ss	Suppresses the output of column statistics in the reload SQL script file.
-t `table`,...	Specifies a list of tables to be unloaded. By default, all tables are unloaded. Together with the -n option, this allows you to unload a set of table definitions only. Table names are always case insensitive, even in case sensitive databases. A `reload.sql` file created with the -t option should not be used to rebuild a database because the file will not include all the database tables. If a table has foreign keys referring to it, the database cannot be rebuilt without the contents of the table. Since the unload file generated does not include database schema users, views, and procedures, the file cannot be used to rebuild a database. Only use the -t option with the -d option to unload data for the tables identified by -t.
-u	Prevents an index from being used to order data. Use this option if you are unloading a database with a corrupt index, so that the corrupt index is not used to order the data. Normally, the data in each table is ordered by the primary key or clustered index if one is defined for the table.
-up	Unloads user passwords (hashed value). You do not need to specify this option if you are performing an unload with reload (-ac, -an, or -ar option). An error is returned if this option is specified with the -no option.

Option	Description
-v	Displays the name of the table being unloaded, and the number of rows that have been unloaded. This option is available only when you run dbunload at a command prompt.
-xi	Performs an external unload by unloading data to the dbunload client, and then using the LOAD statement in the generated reload script file, `reload.sql`, to repopulate the database with data.
-xx	Performs an external unload by unloading data to the dbunload client, and then using the Interactive SQL INPUT statement in the generated reload SQL script file, `reload.sql`, to repopulate the database with data.
-y	Replaces existing script files without prompting for confirmation. If you specify -q, you must also specify -y or the unload will fail if dbunload detects that a script file already exists.
`directory`	Specifies the directory where the unloaded data is to be placed. The directory is relative to the current directory of the database server.

Privileges

For an unload with reload into a new database (-an, -ao, -aob), you must have the SELECT ANY TABLE, SERVER OPERATOR, and ACCESS USER PASSWORD system privileges. For an unload with reload into an existing database (-ac) you do not need the SERVER OPERATOR system privilege.

For an unload without reload, you must have the SELECT ANY TABLE system privilege.

To unload passwords (-up option), you must have the ACCESS USER PASSWORD system privilege.

Remarks

Passwords are only unloaded by default when you are performing an unload with reload (-ac, -an, -ar options). Otherwise, specify the -up option to unload passwords.

Ensure that there are no active transactions (including DDL) occurring on the database when you run the dbunload utility.

When dbunload begins processing, it disables all further logins to the database server and to all databases that may have been started.

If an event occurs during the unload which requires that a login be made to the database server, the login will fail. For example, if an external environment is started by the event then it will attempt to connect to the database server and this will fail.

Once processing is complete, dbunload will re-enable logins.

Unless you include the -up option, a reload script file (default, reload.sql) is generated such that users are created without passwords (for example, GRANT CONNECT,DBA,RESOURCE TO "DBA"). This is a departure from earlier versions of SQL Anywhere. For increased security, now you must explicitly request passwords to be included in the reload script file.

If you do not specify -up, the reload script file contains syntax that makes it unusable for rebuilding your database. If you remove this syntax by editing the script and attempt to use the reload script to recreate the database, the number of database administrators would drop to 0 and you would not be able to connect to the database. The server recognizes this situation and issues the error "Cannot perform specified operation, number of administrators for role 'SYS_AUTH_DBA_ROLE' falls below min_role_admins option value".

When using dbunload with a version 10.0.0 or later database, the version of dbunload used must match the version of the database server used to access the database. If an older version of dbunload is used with a newer database server, or vice versa, an error is reported.

With the Unload utility, you can unload a database and put a set of data files in a named directory. The Unload utility creates a SQL script file that can be used to rebuild your database, provided that you have specified the -up option to include user passwords. It also unloads all the data in each of your tables into files in the specified directory, in comma-delimited format. Binary data is properly represented with escape sequences.

An internal unload/reload unloads information about the current status of each user by issuing UPDATE ISYSUSER statements. An external unload/reload does not include this information and the status of all users is reset.

When you rebuild a database by unloading and reloading it, the rebuilt database may be smaller than the original database. This decrease in database size may be the result of indexing changes, and does not indicate a problem or a loss of data.

If you rebuild a version 11 or earlier database using the Unload utility (dbunload), the new database has global checksums enabled by default, even if the original database had checksums turned off. Although it is not recommended, you can disable global checksums for a database using the ALTER DATABASE statement.

> **i Note**
>
> Version 9 and earlier databases that require recovery cannot be reloaded with version 10 or later of the Unload utility (dbunload). You must reload the database with version 9 or earlier of dbunload.

You can also use the Unload utility to directly create a new database from an existing one. This avoids potential security problems with the database contents being written to ordinary disk files.

There are special considerations for database extraction (unloading) for remote databases involved in replication.

The -gl database server option controls the permissions required to unload data from the database.

The dbo user ID owns a set of system objects in a database, including views and stored procedures.

The Unload utility does not unload the objects that were created for the dbo user ID during database creation. Changes made to these objects, such as redefining a system procedure, are lost when the database is unloaded. Any objects that were created by the dbo user ID since the initialization of the database are unloaded by the Unload utility, and so these objects are preserved.

When you unload a database, changes to permissions on system objects are not unloaded. You must grant or revoke these permissions in the new database.

→ Tip

Before rebuilding your database, validate the reload process by reloading the database without any data, by running a command similar to the following:

```
dbunload -n -an new.db -c "UID=your-user-id;PWD=your-password;DBF=original-database-file"
```

Fix any problems that are identified in the original database before rebuilding it.

In the default mode, or if -ii or -ix is used, the directory used by dbunload to hold the data is relative to the database server, not to the current directory of the user.

If -xi or -xx is used, the directory is relative to the current directory of the user.

If no list of tables is supplied, the whole database is unloaded. If a list of tables is supplied, only those tables are unloaded.

Unloaded data includes the column list for the LOAD TABLE statements generated in the `reload.sql` file. Unloading the column list facilitates reordering of the columns in a table. Tables can be dropped or recreated, and then repopulated using `reload.sql`.

The LOAD TABLE statements generated by dbunload turn off check constraints and computed columns.

Exit codes are 0 (success) or non-zero (failure).

Refresh the materialized views in your database after rebuilding the database.

Tracing information is not unloaded as part of a database unload or reload operation. To transfer tracing information from one database to another, you must do so manually by copying the contents of the sa_diagnostic_* tables; however, this practice is not recommended.

Internal versus external unloads and reloads

The following options offer combinations of internal and external unloads and reloads: -ii, -ix, -xi, and -xx. A significant performance gain can be realized using internal statements (UNLOAD/LOAD) versus external commands (Interactive SQL INPUT and OUTPUT statements). However, internal statements are executed by the database server so that file and directory paths are relative to the location of the database server. Using external commands, file and directory paths are relative to the current directory of the user.

You can specify whether to unload relative to the database server or client using the UNLOAD statement.

When you use an external unload and reload to unload, reload, or rebuild a database and the character set of the database is incompatible with the character set of the host system on which dbunload is running, character set conversion may cause data to be corrupted as it is converted between the database character set and the host system's character set.

To avoid this problem, specify the database character set in the connection string for the database (-c and -ac options). For example, if the database character set is UTF-8, include "charset=utf-8" in the connection strings:

```
dbunload -c UID=user-ID;PWD=password;
CHARSET=utf-8;DBF=filename -ac UID=user-ID;
PWD=password;CHARSET=utf-8;Host=host-name -xx
```

When you perform an external unload, the beginning of `reload.sql` includes a commented CREATE DATABASE statement. This statement can be used to create a database that is equivalent to the one being unloaded.

If the unloaded database was created with version 9 or earlier of SQL Anywhere and had a custom collation, the COLLATION clause appears as follows:

```
COLLATION collation-label DEFINITION collation-definition
```

where `collation-definition` is a string that specifies the custom collation.

The only way to preserve a custom collation is to rebuild the database in a single step (internal unload). If you choose to unload the database, and then load the schema and data into a database that you create, then you must use one of the supplied collations.

If the unloaded database was created with strong encryption, the value of the KEY clause in the CREATE DATABASE statement appears as three question marks (*???*).

Failed unloads

If the -ae option was specified and a failure occurs during the build of a database using one of the -ac, -an or -ar options, dbunload leaves behind a file named `unprocessed.sql` in the current directory. The -ru option changes the path and file name of this file. The file contains all the SQL statements that failed execution. If the file cannot be written to the path, then it is written to a SQL Anywhere temporary directory (refer to -dt).

This file gives you the opportunity to correct, remove, or alter the failing statements. Although the `unprocessed.sql` file is created at the start of processing, it is only written if the -ae option is used. Using Interactive SQL, you can connect to the new database and run the updated `unprocessed.sql` file. This process allows you to complete the database build without starting over again, which can save considerable time.

When the `unprocessed.sql` file contains statements, dbunload returns a failure error code to allow other tools or scripts to be aware of the failed rebuild.

If a failure occurs during an internal rebuild of a database using -ar, then the new database and transaction log file have the `.dbr` and `.logr` file extensions, respectively. Use the following steps to apply the `unprocessed.sql` file and finish the reload manually:

1. Start the new database.
2. Apply the updated `unprocessed.sql` file.
3. Shut down the database.
4. Move `original-name.db` and `original-name.log` to a new directory.
5. Rename the `original-name.dbr` and `original-name.logr` files to `original-name.db` and `original-name.log` respectively.
6. Run the following command:

   ```
   dblog -t original-name.log original-name.db
   ```

Encrypted databases

When you rebuild a database that has table encryption enabled, you must specify either -er or -et to indicate whether the new database has table encryption enabled, otherwise you get an error when attempting to load the data into the new database.

To unload a strongly encrypted database, you must provide the encryption key. You can use the DatabaseKey (DBKEY) connection parameter to provide the key in the command. Or, to be prompted for the encryption key rather than entering it in plain view, you can use the -ep database server option as follows:

```
dbunload -c "DBF=myenc.db;START=dbeng17 -ep"
```

If you are using the -an option to unload a database and reload into a new one, and you want to use the -ek or -ep options to set the encryption key for the new database, keep the following in mind:

- If the original database is strongly encrypted, you need to specify the key for the original database using the DatabaseKey (DBKEY) connection parameter in the -c option, rather than using -ek or -ep.
- Using the -ek and -ep options, it is possible to unload an unencrypted database and reload into a new, strongly encrypted database. When you use -ep and -an, you must confirm the key correctly or the unload fails.
- If the original database is strongly encrypted, but the -ek and -ep options are not used, then the new database will be encoded with simple obfuscation.
- The -ek and -ep options are ignored if -an is not specified. The dbunload -ek and -ep options apply to a new database, while the database server (dbeng17/dbsrv17) options and DBKEY= apply to existing databases.
- When rebuilding databases involved in synchronization or replication, dbunload assumes that the encryption key specified with the -ek or -ep option is the encryption key of the original database, and the encryption key of the newly rebuilt database.

When encrypting the new database or enabling table encryption in the new database, specify an encryption key. The software uses Password-Based Key Derivation Function #2 (PBKDF2), which is part of the PKCS#5 standard to protect the key from brute-force attacks. The software repeatedly applies a cryptographic hash to the encryption key. Use the -kdi option to specify the number of times to apply the hash.

Rebuilding a database

To unload a database, use the dbunload -c option to specify the database name and location with the DatabaseFile (DBF) connection parameter and the DBA user and password with the UserID (UID) and Password (PWD) connection parameters, and include the -up option to include user passwords in the reload script.

To reload a database, create a new database and then run the generated `reload.sql` script file through Interactive SQL.

To combine the unload and reload steps, follow the directions for unloading above, but add the -an option to specify the name of the new database file. See the descriptions of the -ac and -an options.

Example

The following example unloads a database into a `dbreload.sql` file. The CREATE INDEX and LOAD TABLE statements are grouped inside BEGIN PARALLEL WORK statements.

```
dbunload -c "DBF=mydemo.db;UID=DBA;PWD=sql" -up -bp -r dbreload.sql
```

Related Information

Simple Obfuscation Versus Strong Encryption [page 1698]
Diagnostic Tracing (Deprecated) [page 1510]
Reloading Tables with AUTOINCREMENT Columns
Login Policies [page 640]
Corruption Detection Using Checksums [page 993]
Remote Database Extraction
Tips on Exporting Data with the Unload Database Wizard
SQL Anywhere Server Upgrades
Performing a Database Rebuild with Minimum Downtime Using dbunload -ao
Performing a Database Rebuild with Minimum Downtime Using dbunload -aob
Refreshing a Materialized View Manually
SYSTABCOL System View
Software Component Exit Codes
UNLOAD Statement
DatabaseKey (DBKEY) Connection Parameter [page 70]

1.7.4.30 Upgrade Utility (dbupgrad)

Updates the system tables and views, adds new database options, recreates all system stored procedures, installs jConnect support, archives the transaction log, and creates a new transaction log.

> **i** Note
>
> You cannot use the Upgrade utility (dbupgrad) to upgrade SQL Anywhere 9.0.2 and earlier databases to SQL Anywhere 17. To upgrade SQL Anywhere 9.0.2 and earlier databases to SQL Anywhere 17, you must rebuild the database by performing an unload and reload.

This utility also stops and restarts the database.

> Syntax
>
> *dbupgrad* `[ options ]`

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
-c `"keyword=value;..."`	Specifies connection parameters. For example, the following command upgrades a database called sample17 and does not install jConnect support, connecting as user DBA with password passwd: `dbupgrad -c "UID=DBA;PWD=passwd;DBF=c:\sa17\sample17.db"`
-i	Excludes jConnect metadata support. To use the jConnect JDBC driver to access system catalog information, do not use this option. You can still use jConnect when this option is specified, as long as you do not access system catalog information. You can add jConnect metadata support later using SQL Central or the ALTER DATABASE UPGRADE statement.
-nrs	Stops but does not restart the database after the upgrade. If this option is not specified, the database is restarted after the upgrade, unless the AutoStop (ASTOP) connection parameter is set to Yes.
-o `filename`	Writes output messages to the specified file.

Option	Description
-pd [*Y* | *N*]	Specifies whether pre-16.0 system procedures that perform privileged operations execute with the privileges of the definer (owner) or the invoker. When -pd N is specified, the system procedures are executed with the privileges of the invoker (the user who is executing the system procedure). When -pd Y is specified, the system procedures are executed with the privileges of the definer (the user or role that owns the system procedures). If nothing is specified, the current setting for the database that is being upgraded is maintained. In the case of upgrading a pre-16.0 database, this means these system procedures are executed with definer privileges. This setting does not impact system procedures added in version 16.0 or later, and does not impact the default behavior when executing user-defined procedures (executed with the privileges of the definer, unless otherwise specified).
-q	Runs in quiet mode--messages and windows are not displayed.

Privileges

You must have the ALTER DATABASE system privilege, and must be the only connection to the database.

Remarks

> ⚠ Caution
>
> Back up your database files before upgrading.

The dbupgrad utility upgrades a database created with earlier versions of the software to enable features from the current version of the software. The earliest version that can be upgraded is SQL Anywhere 10.0.0. While later versions of the database server can run against databases that were created with earlier releases of the software, some of the features introduced since the version that created the database are unavailable unless the database is upgraded.

An error message is returned if you use the Upgrade utility to upgrade a database that is currently being mirrored.

> **i Note**
>
> Refresh the materialized views in your database after upgrading the database.

> **i Note**
>
> Features that require a physical reorganization of the database file are not made available by dbupgrad. Such features include index enhancements and changes in data storage. To obtain the benefits of these enhancements, you must unload and reload your database.

Upgrading a database does not require you to unload and reload your database.

During the upgrade, the transaction log is archived. A new transaction log is created when the database starts after the upgrade.

Exit codes are 0 (success) or non-zero (failure).

User-defined upgrades are not supported by the Upgrade utility (dbupgrad).

Related Information

Procedures and Functions Running with Owner or Invoker Privileges
Running Pre-16.0 System Procedures as Invoker or Definer
Configuration Files [page 1117]
Alphabetical List of Connection Parameters [page 45]
Database Backup and Recovery [page 939]
SQL Anywhere Server Upgrades
Upgrades and Rebuilds in a Database Mirroring System
User-defined Upgrades
Distributing Privileges Granted to the UPGRADE ROLE System Privilege After an Upgrade [page 1559]
Determining the Security Model Used by a Database (SQL)
Installing jConnect System Objects into a Database
Refreshing a Materialized View Manually
File Hiding Utility (dbfhide) [page 1160]
ALTER DATABASE Statement
Software Component Exit Codes

1.7.4.31 Validation Utility (dbvalid)

Validates the indexes and keys on tables and materialized views.

> **Syntax**

```
dbvalid [ options ] [ object-name, ... ]
```

Option	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, you can use the File Hiding utility (dbfhide) to encode the contents of the configuration file.
-c `"keyword=value;..."`	Specifies database connection parameters. For example, the following command validates the database `c:\salesdata.db`, including tables and materialized views: `dbvalid -c "UID=DBA;PWD=passwd;DBF=c:\salesdata.db"`
-d	Validates that all pages in the database belong to the correct object, and performs a checksum validation. The -d option validates the correctness of indexes. The -d option cannot be used with the -i, -s, or -t options.
-fx	Disables primary key/foreign key referential integrity checking. The -fx option cannot be used with the -i option.
-i	Defines `object-name` as a list of indexes.

Option	Description
-im `mode`	Defines the in-memory mode to use if the database server is auto-started (by setting the -im database server option on the StartLine parameter of the connection string). Mode is one of the following: **none** In-memory mode is not used. **c** Checkpoint only. This mode is useful in applications where increased performance is desirable and the loss of committed transactions after the most recent checkpoint is acceptable (when you run the database server in checkpoint only mode, the database server does not perform a checkpoint after each commit). **nw** Never write. Committed transactions are not written to the database file on disk. All changes are lost if the database is shut down or crashes, so database files are always left in their original state. **v or validation (default)** In-memory validation. Use this mode to validate a backup copy of the database without modifying it or its transaction log files.
-o `filename`	Appends output messages to the named file.
-q	Does not display output messages to the client. You can still log the messages to file using the -o option.
-s	Validates the database using checksums. Checksums are used to determine whether a database page has been modified on disk. Checksum validation reads each page of the database from disk and calculates its checksum, if the page has a checksum. If the calculated checksum is different from the checksum stored on the page, then the page has been modified on disk and an error is returned. The page numbers of any invalid pages appear in the database server messages window. The -s option cannot be used with the -d, -i, -t, or -fx options.
-t	Defines `object-name` as a list of tables and materialized views.
-wl	Runs validation with exclusive data locks applied to the table(s). Transactions can read, but not modify the table data or schema.

Option	Description
-ws	Runs validation with snapshot isolation applied to the table(s). Transactions can read and modify the data. This option requires that the database have snapshot isolation enabled. Because this clause uses snapshot isolation, performance is often affected.
`object-name`	Specifies the name of the table, materialized view, or index to validate. If -i is used, `object-name` refers to an index to validate instead.

Privileges

You must have the VALIDATE ANY OBJECT system privilege. If a specific index is specified, then you must be the owner of the table on which the index is created, or have the VALIDATE ANY OBJECT system privilege.

Remarks

With the Validation utility, you can validate the indexes and keys on some, or all, of the tables and materialized views in a database. You can also use the Validation utility to verify the database file structure to ensure that all pages in the database belong to the correct object, and that page checksums are correct. By default, dbvalid validates all the tables, materialized views, and indexes in the database and validates the database file structure.

When you validate a table, dbvalid also validates all of the table's indexes to verify that the set of rows and values in the table matches the set of rows and values contained in each index. All BLOBs in the table are also traversed, BLOB allocation maps are verified, and orphaned BLOBs are detected. The Validation utility also checks the physical structure of all index pages, the ordering of the index hash values, and the index's uniqueness requirements (if any are specified). Unless the -fx option is specified, each foreign key value is looked up in the corresponding primary key table to check that referential integrity constraints are intact.

When the -i option is specified, dbvalid validates each index in the object list. Validating an index works exactly the same as validating a table, except that only the specified index and its underlying table are validated. If the index is a foreign key, then each value is looked up in the primary key table unless the -fx option is specified.

If you start database validation while the database cleaner is running, then the validation does not run until the cleaner is finished running.

The Validation utility can be used in combination with regular backups to give you confidence in the integrity of the data in your database. To validate a backup copy of your database, make a copy of the backup and validate the copy. This process ensures that you do not make changes to the file that is used in recovery.

⚠ Caution

Backup copies of the database and transaction log must not be changed in any way. If there were no transactions in progress during the backup, or if you specified BACKUP DATABASE WITH CHECKPOINT

LOG RECOVER or WITH CHECKPOINT LOG NO COPY, then you can check the validity of the backup database using read-only mode or by validating a copy of the backup database.

However, if transactions were in progress, or if you specified BACKUP DATABASE WITH CHECKPOINT LOG COPY, the database server must perform recovery on the database when you start it. Recovery modifies the backup copy of the database and subsequent transaction log files from the original database cannot be applied. In this case, use in-memory validation to prevent changing the database and transaction logs or validate a copy of the database.

If running the Validation utility starts a database automatically, then the database starts in read-only mode. This behavior prevents changes from being made to the database if the validation is part of a backup or recovery plan.

If the Validation utility connects to a running database that was not started in read-only mode, then the utility displays a warning. This warning is a reminder that the database being validated cannot be used as part of a recovery plan. Because of the way backups are performed, most databases created by dbbackup are marked as needing recovery. If the database you are validating requires recovery and you want to force it to start as read-write, you can either start the database before running dbvalid or specify a valid value for the DBS connection parameter.

Both of the following commands allow dbvalid to run if the `mycopy.db` database needs to be recovered:

```
dbvalid -c "UID=DBA;PWD=passwd;DBF=mycopy.db;DBS=-n mycopy"
```

```
dbvalid -c "UID=DBA;PWD=passwd;DBF=mycopy.db;DBS=-dh"
```

⚠ Caution

Validation does not acquire exclusive access to the table being validated and may report failures if the table is modified during validation. If the -wl or -ws option is not specified, then only perform validation while no connections are making changes to the table; otherwise, errors may be reported indicating some form of database corruption even though no corruption actually exists.

The Validation utility may return warnings about checksum violations for databases that do not have global checksums enabled. This is because the database server automatically calculates checksums for critical database pages, regardless of whether checksums are enabled. A database may also have checksums on some pages because it was started with write checksums.

The database server creates checksums automatically for databases running on storage media that may be less reliable, such as removable drives.

Exit codes are 0 (success) or non-zero (failure).

Related Information

Configuration Files [page 1117]
Database Backup and Recovery [page 939]
Corruption Detection Using Checksums [page 993]
Validating a Database (SQL Central) [page 990]
File Hiding Utility (dbfhide) [page 1160]

sa_clean_database System Procedure
DatabaseSwitches (DBS) Connection Parameter [page 73]
Software Component Exit Codes
VALIDATE Statement
-im Database Server Option [page 454]

1.7.4.32 Version Diagnostic Utility (dbversion)

Returns information about the specified executable.

Syntax

```
dbversion executable-name
```

Remarks

This utility is only available on Unix, and returns information about SQL Anywhere executables.

Example

The following command:

```
$ dbversion /opt/sqlanywhere17/bin32/dbversion
```

returns information about the dbversion executable:

```
SQL Anywhere Version Diagnostic Utility Version 17.0.11.1293
/opt/sqlanywhere17/bin32/dbversion: dbversion GA 16 0 0 1403 linux 2012/10/10
15:32:50 nothr 32 production
```

Field	Description
dbversion	Returns the executable name.
GA	Returns a two-letter code designating an install type.
17	Returns the major version number.
0	Returns the minor version number.
0	Returns the support package number.
1403	Returns the build number.
linux	Returns the operating system code.
2012/10/10 15:32:50	Returns the build date/timestamp.

Field	Description
nothr	Returns the threading model (nothr or posix).
32	Returns the bitness of the executable (32 or 64).
production	Returns either production or debug.

Related Information

-v Database Server Option [page 520]

1.8 Performance Improvements, Diagnostics, and Monitoring

Read tips on how to monitor and improve database performance, and diagnose problems.

In this section:

SQL Anywhere Monitor, Cockpit, and Profiler Comparison [page 1264]
The Cockpit shows you what your database server is doing, the Monitor shows which database servers are currently available, and the Profiler helps you investigate why your database is slow.

Monitoring [page 1264]
You can use a number of tools to monitor the status of a database and database server.

Performance [page 1419]
You can use a number of tools to improve performance of a database and database server.

Diagnostics [page 1482]
You can use a number of tools to diagnose problems in a database and database server.

Related Information

SQL Anywhere Cockpit [page 1419]
SQL Anywhere Monitor [page 1265]
SQL Anywhere Profiler [page 1483]
SQL Anywhere Profiler [page 1483]

1.8.1 SQL Anywhere Monitor, Cockpit, and Profiler Comparison

The Cockpit shows you what your database server is doing, the Monitor shows which database servers are currently available, and the Profiler helps you investigate why your database is slow.

Cockpit

Use the Cockpit to quickly find out what is currently occurring in a database server and the databases on which it runs, and to receive email notifications when issues arise. In addition to highlighting database server issues, the Cockpit makes it easy to view information about your connected users, database, and database server properties, as well as database server messages.

Profiler

Use the Profiler to troubleshoot and fine tune your applications during development, as well as in production. The Profiler can log all SQL statements running in your database, including those from within stored procedures, triggers, and events. Use the stored SQL profiling feature to compare and benchmark the run times of these stored SQL objects against alternative implementations. Profiling all statements also provides detailed views of blocked and deadlocked connections.

Monitor

Use the SQL Anywhere Monitor to see the health and availability of all your servers in the enterprise, and receive notification emails when issues arise. The Monitor handles multiple databases, database servers, MobiLink servers, and server farms, as well as web servers, proxy servers, and host computers.

Related Information

SQL Anywhere Cockpit [page 1419]
SQL Anywhere Profiler [page 1483]
SQL Anywhere Monitor [page 1265]

1.8.2 Monitoring

You can use a number of tools to monitor the status of a database and database server.

In this section:

1.8.2.1 SQL Anywhere Monitor

The SQL Anywhere Monitor, also referred to as the Monitor, is a browser-based administration tool that provides you with information about the health and availability of SQL Anywhere databases, MobiLink servers, and MobiLink server farms. It can also provide information about the availability of web servers, proxy servers, and host computers.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

The Monitor provides the following functionality:

Constant data collection

Unlike many of the other administration tools, the Monitor collects metrics all the time, even when you are not logged in to the browser. The Monitor collects metrics until you shut it down.

Email alert notification

As the metrics are collected, the Monitor examines the metrics and can send email alerts when it detects conditions that indicate something is wrong with a resource.

Browser-based interface

At any time, you can log in to the Monitor using a browser to review alerts and metrics that have been collected.

Monitor multiple databases, MobiLink servers, MobiLink server farms, web servers, proxy servers, and host computers

From one tool, you can simultaneously monitor multiple resources running on the same or different computers.

Minimal performance impact

The Monitor can be used routinely in development and production environments because monitoring does not degrade performance.

The Monitor is designed to help any type of user, whether they are a database administrator or not, who is responsible for such tasks as:

- Ensuring that a database or a MobiLink server is connected to the network.
- Ensuring that there is enough disk space or memory available for a database or a MobiLink server.
- Ensuring that users aren't blocked or that queries aren't taking too long to run.
- Reviewing the number of synchronizations a MobiLink server performs over a specified time.
- Ensuring that a web server, proxy server, or host computer is available.

In this section:

An **alert** is a condition or state of interest about a resource that should be brought to a Monitor administrator's or operator's attention.

Backing up the Monitor [page 1348]
By default, the Monitor performs maintenance on the metrics once a day at midnight. Maintenance affects metrics, not alerts.

List of Database Objects Installed by the Monitor [page 1351]
There are several objects that are installed when you add a database as a resource to be monitored.

Deleting Monitoring Objects from the Resource Database [page 1352]
Delete the monitoring objects from the resource database that were installed when you added the resource to the Monitor.

Monitor Communications Security [page 1353]
You can secure communications between both the Monitor and your browser, and between the Monitor and the resources it monitors.

Troubleshooting the Monitor [page 1354]
There are several ways Monitor administrators can troubleshoot the Monitor.

Tutorial: Monitoring Resources with the Monitor [page 1356]
Set up monitoring for a database or MobiLink server.

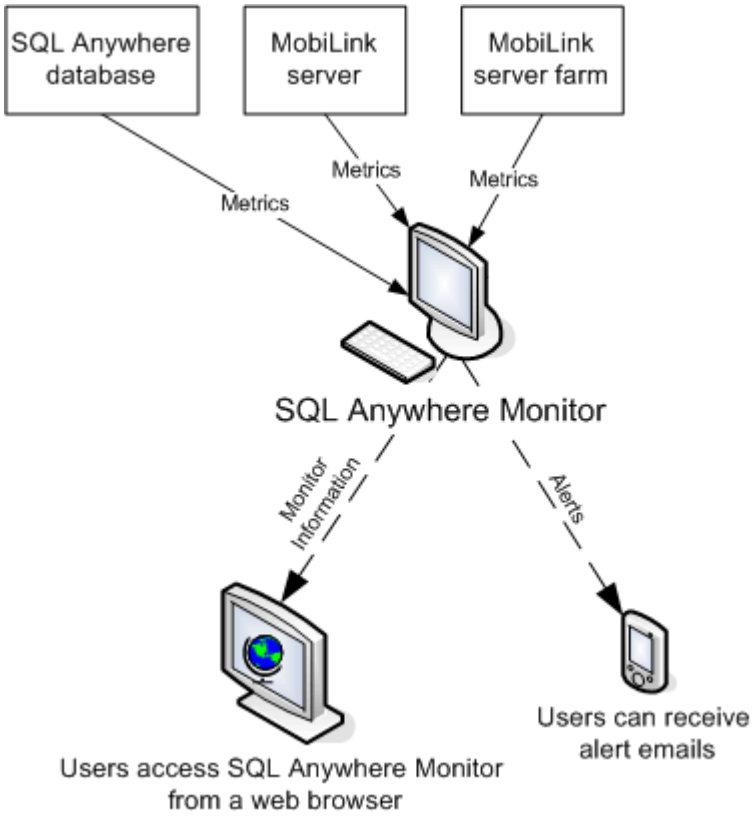
Related Information

SQL Anywhere Profiler [page 1483]
Windows Performance Monitor [page 1506]
MobiLink Profiler
SQL Anywhere Cockpit [page 1419]
Performance Statistics Utility (dbstats) (UNIX/Linux) [page 1193]

1.8.2.1.1 Monitor Architecture

The Monitor collects metrics and performance data from databases (including those involved in read-only scale-out and mirroring systems), MobiLink servers, and MobiLink server farms, while a separate computer accesses the Monitor via a browser. In addition, it can monitor the availability of web servers, proxy servers, and host computers.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.



The Monitor is an application that is served to a web browser via the SQL Anywhere built-in HTTP server.

There are two editions of the Monitor:

SQL Anywhere Monitor Developer Edition

The Monitor Developer Edition is intended for development and testing use. It is included in the installation by default and it uses the installed software on the back end.

SQL Anywhere Monitor Production Edition

The Production Edition is intended for deployment and production use. It is installed separately, runs as a service, and includes a fully contained SQL Anywhere installation. This edition is only available for certain editions of the software.

Requirements

For the computer where the Monitor is installed

- To run the Monitor on a Windows system that has the Windows Firewall enabled, you must add a port exception for port 4950.
- To monitor resources that are secured using FIPS-certified encryption, you need a separate license.
- The Monitor can run on the same computer as the resources it is monitoring, but it is recommended, particularly in production environments, that you run the Monitor on a different computer to minimize the impact on the database server, the MobiLink server, or other applications.

- It is recommended in production environments that you run the Monitor Production Edition.
- When running the Monitor Developer Edition, you must have SQL Anywhere Server 17 installed. The Monitor Developer Edition uses the installed SQL Anywhere on the back end.

For the computer accessing the Monitor

- Install the latest version of Adobe Flash Player that is available for your operating system. The Monitor is backward compatible with version 10 of Adobe Flash Player.
- Enable JavaScript in your browser.
- Ensure that the computer where you are using a browser to access the Monitor is connected to the network where the Monitor is installed.
  You can access the Monitor from the same computer as it is running on, but it is recommended, particularly in production environments, that you access the Monitor from a different computer.

Limitations

- You can only run one edition of the Monitor on a computer at a time.
- On Linux, the Monitor Developer Edition can only be run by the user who installed it.
- On Linux, only the user with a Linux administrator privilege can install and run the Monitor Production Edition.
- The Monitor connects to databases over TCP/IP. To monitor a database running on the local database server (dbeng17), start the database server with the -x all or -x tcpip option.
- You cannot use the Monitor to optimize queries or determine the speed of your application. If you are interested in tuning database and application performance, you can use such tools as the *Profiler* or the *Cockpit*.
- The Monitor does not provide information about individual synchronizations. For detailed information about individual synchronizations, including timing and other per-synchronization statistics, use the MobiLink Profiler.
- For information about the versions of databases, MobiLink servers, and MobiLink server farms that the Monitor supports, see http://scn.sap.com/docs/DOC-63186.

Related Information

About the SQL AnywhereMonitor Production Edition
Monitor Communications Security [page 1353]
MobiLink Profiler
Supported Platforms

1.8.2.1.2 Quick Start to Using the Monitor

Monitor a database, MobiLink server, and MobiLink server farm.

> **i Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

1. Start the database or MobiLink server that you want to monitor (if it is not already running).
2. Install the Monitor on a computer that is always connected to your network.
   The Monitor can run on the same computer as the resources it is monitoring, but it is recommended, particularly in production environments, that you run the Monitor on a different computer to minimize the impact on the resources or other applications.
3. Start the Monitor and open it in your browser.
4. Log in to the Monitor as an administrator. The default user is an administrator with the name **`admin`** and the password **`admin`**.

   > **i Note**
   >
   > You must be logged in to the Monitor as an administrator to perform the following tasks.

5. As a Monitor administrator, in the left navigation menu, click *Tools* > *Administration* and add the resource to be monitored.
6. As a Monitor administrator, add new users and change the password for the admin user.
7. As a Monitor administrator, configure alert thresholds for the resources.
8. As a Monitor administrator, configure the backup and maintenance schedule.
9. Click *Close* to exit the *Administration* window.
10. Click *Overview*. In the *Resource List* widget, the resources that are being monitored appear.

Related Information

Dashboards and Widgets [page 1277]
Monitor Users [page 1331]
Backing up the Monitor [page 1348]
Starting the Monitor [page 1271]
Adding a Database Resource [page 1289]
Adding a MobiLink Server Resource [page 1291]
Creating Monitor Users [page 1333]
Alerts [page 1339]

1.8.2.1.3 Starting the Monitor

Start and log in to the Monitor.

Prerequisites

> **i Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Ensure that there isn't another copy of the Monitor running on the computer.

Context

By default, the Monitor Production Edition runs automatically as a service when installed, and starts when the computer starts. If you stop the Monitor Production Edition, you can restart it using these instructions.

Procedure

1. On the computer where the Monitor is installed, start the Monitor.

   On Windows

   1. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Anywhere Monitor*.
   2. Alternatively, for the Monitor Production Edition, you can run the `samonitor` batch file from the `bin32` or `bin64` directory in the Monitor installation directory to restart the service.

      ```
      samonitor.bat start
      ```

   On Linux

   Run the `samonitor.sh` script from the `bin32` or `bin64` directory in the Monitor installation directory:

   ```
   ./samonitor.sh launch
   ```

   > **i Note**
   >
   > If you are starting the Monitor Developer Edition, the Monitor icon appears in the system tray.

   The Monitor connects to the resources and begins collecting metrics for all resources in the Monitor.
2. Browse to the URL for logging in to the Monitor. The default URL is `http://localhost:4950`.

> **Note**
>
> If you are logging in remotely to the Monitor, browse to http://`computer-name`:4950, where `computer-name` is the name of the computer where the Monitor is running.

3. Log in.

   In the browser, enter your user name and password for the Monitor. The user name and password for the Monitor are case sensitive. There are three different types of users: administrators, operators, and read-only users. Each type of user has different privileges. The default user is a Monitor administrator with the name **`admin`** and the password **`admin`**.

   Click *Remember Me On This Computer* to have your session persist for two weeks or until you log out. If *Remember Me On This Computer* is cleared, your session expires when you close the browser or log out.

Results

The Monitor is started.

Next Steps

The Monitor is designed to run silently in the background. You interact with the Monitor through its browser interface. It is recommended that you leave the Monitor running continuously in the background to collect metrics.

Related Information

Monitor Users [page 1331]
Dashboards and Widgets [page 1277]
Stopping the Monitor [page 1273]
Logging out from the Monitor [page 1275]
Logging in Remotely to the Monitor [page 1274]

1.8.2.1.4 Stopping the Monitor

Stop the Monitor Production Edition by stopping the Monitor service and the collection of metrics.

Context

> **i Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

> **⚠ Caution**
>
> In most cases, it is recommended that you leave the Monitor running, but close the browser. Closing the browser does not stop the collection of metrics.

You can also stop monitoring a specific resource such as a database

Procedure

Stop the Monitor.

On Windows

Choose one of the following options:

Option	Action
Monitor Developer Edition	In the system tray, right-click the Monitor icon and click *Exit SQL Anywhere Monitor*.
Monitor Production Edition	Run `samonitor.bat` from the `bin32` directory in the Monitor installation directory: `samonitor.bat stop`

On Linux

Choose one of the following options:

Option	Action
Monitor Developer Edition	`./samonitor.sh stop`
Monitor Production Edition	`samonitor.sh stop`

Run the `samonitor.sh` script from the `bin32` or `bin64` directory in the Monitor installation directory:

Results

The Monitor is stopped, and the Monitor stops collecting metrics for all resources.

Related Information

Dashboards and Widgets [page 1277]
Stop Monitoring Resources [page 1301]
Starting the Monitor [page 1271]
Logging in Remotely to the Monitor [page 1274]
Logging out from the Monitor [page 1275]

1.8.2.1.5 Logging in Remotely to the Monitor

Log in remotely to the Monitor, by browsing to the URL for the Monitor.

Prerequisites

The computer that you are using to log in to the Monitor must be connected to the network where the Monitor is running.

Context

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were

previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Procedure

1. On the computer where the Monitor is installed, start the Monitor.
2. On the computer that is accessing the Monitor, browse to the default URL for logging in to the Monitor: http://`computer-name`:4950, where `computer-name` is the name of the computer the Monitor is running. For example, `http://samplehost:4950`.
3. When prompted, enter your user name and password for the Monitor. The user name and password for the Monitor are case sensitive.

Results

You are logged in to the Monitor from the remote computer.

Related Information

Monitor Users [page 1331]
Dashboards and Widgets [page 1277]
Starting the Monitor [page 1271]
Stopping the Monitor [page 1273]
Logging out from the Monitor [page 1275]

1.8.2.1.6 Logging out from the Monitor

Log out from the Monitor.

Context

Logging out from the Monitor has *no* effect on the collection of metrics.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were

previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Procedure

Click *Logout* in the upper right corner.

If you click *Remember Me On This Computer* when you log in to the Monitor, then closing the browser does not log you out of the Monitor.

Results

You are logged out of the Monitor.

Related Information

Stop Monitoring Resources [page 1301]
Dashboards and Widgets [page 1277]
Starting the Monitor [page 1271]
Stopping the Monitor [page 1273]
Logging in Remotely to the Monitor [page 1274]

1.8.2.1.7 Customizing Accessibility Features

Change the font size and color theme in the Monitor.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Procedure

1. Click *User Settings*.
2. Choose one of the following options:

Option	Action
Change the font size	Select the new font size from the *Font Size* dropdown list.
Change the color theme	Choose one of the following options: • Normal • High Contrast (black) • High Contrast (white)

3. Click *Save*.

1.8.2.1.8 Dashboards and Widgets

The *Overview* dashboard provides an overview of the health and availability of the resources being monitored. By default, the *Overview* dashboard contains the *Resource List* widget and the *Alert List* widget.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Resource List widget

The *Resource List* widget contains a table that lists the resources being monitored, and provides as an indication about the overall health of each resource. The table always contains the default resource, named *SQL Anywhere Monitor*, which reports on the health of the Monitor itself. You cannot modify the *SQL Anywhere Monitor* resource, nor can you stop monitoring it.

To view detailed information about a resource, click the resource name in the *Resource List* widget, which opens the dashboard for the selected resource.

In the *Resource List* widget, the *Status* column provides information about the connections between the Monitor and its resources. The *Status* column indicates whether the resource requires someone to perform an action on it.

A resource has one of the following statuses:

Icon	Status	Description
No icon present	*Healthy*	There are no unresolved alerts for the resources.
	*Alerts Present*	There are one or more unresolved alerts for the resource.
	*Unavailable*	The resource is unavailable. For example, the resource is down.
	*Monitoring Stopped*	The resource is not being monitored because of a blackout or because a user manually stopped monitoring the resource.

Alert List widget

The *Alert List* widget contains the alerts for the monitored resources. To view detailed information about an alert, click the alert in the *Alert List*, and then click *Details*, which opens a window showing the details of the alert.

An alert has one of the following statuses:

Icon	Status	Description
	*Active*	The alert condition still applies. No one has resolved the alert.
	*Inactive*	The issue that triggered the alert is no longer present, but the alert has not been resolved or deleted.
	*Resolved*	A Monitor administrator or operator has marked the alert resolved.

An alert has one of the following levels of severity:

Icon	Severity	Description
	High severity	High severity alerts indicate problems that require a user's immediate attention. For example, when a resource exceeds the low disk space threshold, a high severity alert is issued.

Icon	Severity	Description
	Medium severity	Medium severity alerts indicate problems that require a user's attention as the problems could escalate. For example, when a resource exceeds the CPU usage threshold, a medium severity alert is issued.
	Low severity	Low severity alerts indicate problems. For example, when a resource has a failed connection, a low severity alert is issued.

In this section:

Related Information

Alerts [page 1339]

1.8.2.1.8.1 Managing Widgets

Add, edit or delete a widget from the dashboard.

Context

The following table contains the different type of metric widgets you can create for your resources:

Type of metric	Description
Key Performance Metrics	Displays information gathered for a resource you are monitoring. You can choose to view metrics as either a graph or a table.
Alerts	Displays a list of alerts for the resources.
Resources	Displays a list of resources.
SQL Anywhere Connections	Displays a list of database connections.
SQL Anywhere Scale-Out Topology	Displays a topology of mirroring and scale-out systems.

Procedure

Choose one of the following options.

Option	Action
Add a widget	1. Open the dashboard. 2. In the upper right corner of the dashboard, click *Customize*, and then click *Add Widget*. 3. Click the type of widget that you want for the *What type of widget do you want* field, and then click *Next* 4. Specify a name for the widget in the *What do you want to name this widget* field. 5. Specify the resource in the *Which resource are you interested in* field. 6. Follow the instructions in the wizard. 7. Click *Create*.
Edit a widget	1. In the widget, click the dropdown arrow in the upper right corner of the widget. 2. Edits the widget's settings.
Delete a widget	1. In the widget, click the dropdown arrow in the upper right corner of the widget. 2. Deletes the widget.

1.8.2.1.8.2 *SQL Anywhere Scale-out Topology* Widget

You can view the topology of read-only scale-out and mirroring systems by creating a *SQL Anywhere Scale-Out Topology* widget.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Viewing a read-only scale-out system in a topology widget

The *SQL Anywhere Scale-Out Topology* widget for a read-only scale-out system displays the following fields and their possible values:

Server Name

The name of the node in the scale-out system.

Type

The server type. The type is either *Partner - Primary* or *Copy*, depending on how the server was defined when the read-only scale-out system was created.

State

The connection status of the node. Can be one of *Connected* or *Disconnected*.

Connections

The current number of connections to this node.

Last Updated

The last time the server updated its status.

Viewing a database mirroring system in a topology widget

The *SQL Anywhere Scale-Out Topology* widget for a database mirroring system displays the following fields and their possible values:

Server Name

Displays the names of the database servers in the mirroring system.

Type

Lists the server type and displays one of the following values:

Partner - Primary

This value identifies the server that is currently acting as the primary.

Partner - Mirror

This value identifies the server that is currently acting as the mirror.

Arbiter

This value identifies the arbiter server.

State

Displays the connection status of each server.

Connections

Shows the number of current connections to the server.

Last Updated

Displays the last time the server's status was updated.

Related Information

Database Mirroring [page 1757]
Read-only Scale-out [page 1830]
SQL Anywhere Monitor [page 1265]
Tutorial: Monitoring Resources with the Monitor [page 1356]
Tutorial: Creating a Read-only Scale-out System [page 1853]
Tutorial: Creating a Database Mirroring System [page 1800]
CREATE MIRROR SERVER Statement
sa_mirror_server_status System Procedure

1.8.2.1.8.3 Managing Dashboards

Add, edit, and delete a dashboard, or create a template from an existing dashboard.

Context

> **i Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Dashboards are specific to each user. In a template dashboard, the user specifies the default layout and widgets that appear in new dashboards. A dashboard template cannot be set on the *SQL Anywhere Monitor* resource.

Procedure

1. Log in to the Monitor.
2. Choose one of the following options:

Option	Action
Add a dashboard	1. In the *Dashboards* pane, click *Add New*. 2. Follow the instructions in the window to add a dashboard.
Edit a dashboard	1. Open the dashboard. 2. In the upper right corner of the dashboard, click *Customize*, and then click *Settings*.
Delete a dashboard	1. Open the dashboard. 2. In the upper right corner of the dashboard, click *Customize*, and then click *Delete*.
Create a dashboard template	1. Open the dashboard that you want to use as the template. 2. In the upper right corner of the dashboard, click *Customize*, and then click *Set As Template*.

Related Information

Managing Widgets [page 1280]
Managing Widgets [page 1280]

1.8.2.1.8.4 Closing a Connection Using the Monitor

View and close connections to database resources using the Monitor.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must know the user name and password of a user of the database who has the DROP CONNECTION system privilege.

Procedure

1. Log in to the Monitor.
2. Open the dashboard for the database resource.

   Click *Dashboards* > `name-of-dashboard`.
3. In the *Connections* widget, select a connection, and then click the *x* beside the connection name.
4. When prompted, type the user name and password for the database.

   These credentials are used to connect to the database and close the specified connection. The credentials are then removed from the Monitor.
5. Click *OK*.

1.8.2.1.8.5 Understanding How Time Is Displayed

Use the *Server Time Offset* metric to determine the offset for the time being displayed.

Context

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Displayed times in the Monitor are based on the 24-hour clock and are local to the time on the computer that the Monitor is running on.

Procedure

1. Log in to the Monitor.
2. Open the dashboard for the resource.
3. In the *Server Info* widget, find the *Server Time Offset* metric.

Results

The *Server Time Offset* records the time difference between the time on the computer that the Monitor is running and the time on the computer that you are using to view the Monitor data.

Related Information

Metrics [page 1307]

1.8.2.1.9 *Administration* Window for the Monitor

As a Monitor administrator, you can select the resources (for example, databases, MobiLink servers and MobiLink Server Farms) that you want to monitor.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

> **i** Note
>
> Only Monitor administrators can access the *Administration* window.

As a Monitor administrator, you can also:

- Add, edit, and delete resources.
- Add and edit users.
- Backup and configure the Monitor.
- View the Message Log.
- View the Exception Reports.

In this section:

Viewing the Message Log [page 1286]
View the Message Log using the table it appears in, with the most recent message at the top.

Viewing Exception Reports [page 1287]
View an exception report that details what happened at the time of the problem when the Monitor experiences a fatal error or a crash.

Related Information

Monitor Users [page 1331]
Backing up the Monitor [page 1348]
Metrics [page 1307]

1.8.2.1.9.1 Viewing the *Message Log*

View the Message Log using the table it appears in, with the most recent message at the top.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

Context

The *Message Log* contains informational messages from and about the Monitor regarding its operation.

Procedure

1. Log in to the Monitor.
2. Click *Tools* > *Administration*.
3. Click *Message Log*.

Results

You can view the *Message Log*.

1.8.2.1.9.2 *Viewing Exception Reports*

View an exception report that details what happened at the time of the problem when the Monitor experiences a fatal error or a crash.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

Procedure

1. Log in to the Monitor.
2. Click *Tools* > *Administration*.
3. Click *Exception Reports*.

Results

You can view the exception report.

1.8.2.1.10 Resources

A **resource** is a database (including databases in read-only scale-out or mirroring systems), a MobiLink server, a MobiLink Server Farm, or a web service.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

As a Monitor administrator, you add resources to the Monitor, and then you start monitoring them.

SQL Anywhere Monitor **Resource**

The default resource, named *SQL Anywhere Monitor*, reports on the health of the Monitor itself. This default resource is useful for monitoring the computer that the Monitor is running on and sending alerts when the Monitor is experiencing problems. You cannot modify this resource, nor can you stop monitoring it.

In this section:

Related Information

Dashboards and Widgets [page 1277]

1.8.2.1.10.1 Adding a Database Resource

Add a database (including databases in read-only scale-out and mirroring systems) as a resource to the Monitor.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

As of version 16, to add a database, you need the user ID and password for a user in the database that has the SYS_SAMONITOR_ADMIN_ROLE role. To add an earlier database, you need the user ID and password for a user in the database that has DBA authority.

Context

You can add a resource one at a time or you can add multiple resources via a CSV file.

There is no difference in the resource configuration for a root node in a read-only scale-out system and other database servers. When the resource is configured, the Monitor detects that the resource is the root node in a read-only scale-out system.

Procedure

1. Log in to the Monitor.
2. Click *Administration*.
3. Click *Resources*.
4. Click *Add*.
5. In the *Add Resource* window:
   a. Click *SQL Anywhere Server*.
   b. Click *Next*.
   c. In the *Name* field, type a name for the resource. This is the name that appears in the Monitor. (Click *Next*).
   d. In the *Host* field, specify the hostname or IP address of the computer on which the database server is running. You can use the name localhost to represent the current computer. This value is required.

If you are monitoring the primary database in a mirroring system, then specify the host names and port numbers of the computers running the primary and mirror servers in a comma-separated list. For example: `my-primary-server:2638,my-mirror-server:49152`.

e. (Optional) In the *Port* field, specify the TCP/IP port that the database is running on.

If you are monitoring the primary database in a mirroring system, then ensure that the *Port* field is empty.

f. (Optional) In the *Databases* field, specify the database name.

g. (Optional) In the *Server* field, specify the server name.

If you are monitoring the primary database in a mirroring system, then in the *Server* field, type the alternate server name for the primary server. Specify the name that clients use to connect to the database server that is acting as the primary server in the database mirroring system.

h. (Optional) In the *Other* field, specify a semicolon-delimited string of connection parameters.

i. Click *Next*.

j. Follow the remaining instructions.

k. Click *Create*.

6. When prompted, enter the user ID and password of a user that has the SYS_SAMONITOR_ADMIN_ROLE role or DBA authority, and then click *Close*.

These credentials are used to:

- Connect to the database.
- Create a new user named sa_monitor_user. The Monitor uses the sa_monitor_user to connect to the database and monitor it.
- Install the database objects needed by the sa_monitor_user to monitor the database.
- Discard the user credentials from the Monitor. Because the sa_monitor_user is added to the database being monitored, it is not necessary to store the user credentials outside the database that is being monitored.

7. Click *Close*.
8. A prompt window appears with the option to create a web service resource to monitor. To configure a web service resource to be monitored, click *Yes*; otherwise, click *No*.
9. Click *Close*.

Results

The resource is added and monitoring of the resource starts automatically. The resource appears in the *Overview* dashboard *Resource List*.

Next Steps

Optionally, you can add a dashboard for the resource. By default, when a resource is added a dashboard is not.

You can add a widget to your dashboard. To view the topology of read-only scale-out and mirroring systems, you must add a *SQL Anywhere Scale-Out Topology* widget.

Related Information

SQL Anywhere Scale-out Topology Widget [page 1281]
MobiLink Users in a Synchronization System
Adding Multiple Resources with a CSV File [page 1295]
Managing Widgets [page 1280]
Managing Dashboards [page 1282]
Starting the Monitor [page 1271]
Host Connection Parameter [page 86]
List of Database Objects Installed by the Monitor [page 1351]

1.8.2.1.10.2 Adding a MobiLink Server Resource

Add a MobiLink server as a resource to be monitored in the Monitor.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

Context

You can add a resource one at a time or you can add multiple resources via a CSV file.

To monitor a pre-version 16 MobiLink server that uses ECC encryption, you must specify a second set of network protocol options that does not use ECC when starting the MobiLink server.

The Monitor does not support ECC encryption. To monitor a pre-version 16 MobiLink server that uses ECC encryption, you must specify two port numbers when starting the MobiLink server. When prompted in the Monitor to specify the port number for the MobiLink server resource, specify the number that is not associated with ECC encryption.

Procedure

1. Log in to the Monitor.
2. Click *Administration*.
3. Click *Resources*.
4. Click *Add*.
5. In the *Add Resource* window:
   a. Click the type of resource that you want to monitor.
   b. Click *Next*.
   c. In the *Name* field, type a name for the resource. This is the name that appears in the Monitor, and then click *Next*.
   d. In the *Host* field, specify the hostname or IP address of the computer on which the MobiLink server is running. You can use the name localhost to represent the current computer. This value is required.
   e. Click *Next*.
   f. Follow the remaining instructions.

      For MobiLink, you must supply a MobiLink user ID and password that are used to connect to the MobiLink server. The MobiLink user ID and password are stored in the Monitor.
   g. Click *Create*.
6. Click *Close*.
7. Click *Close*.

Results

The resource is added and monitoring of the resource starts automatically. The resource appears in the *Overview* dashboard *Resource List*.

Next Steps

Optionally, you can add a dashboard for the resource. By default, when a resource is added a dashboard is not.

You can add a widget to your dashboard. To view the topology of read-only scale-out and mirroring systems, you must add a *SQL Anywhere Scale-Out Topology* widget.

Related Information

SQL Anywhere Scale-out Topology Widget [page 1281]
MobiLink Users in a Synchronization System
Adding Multiple Resources with a CSV File [page 1295]
Managing Dashboards [page 1282]

Starting the Monitor [page 1271]
Managing Widgets [page 1280]
Host Connection Parameter [page 86]
List of Database Objects Installed by the Monitor [page 1351]

1.8.2.1.10.3 Adding a MobiLink Server Farm Resource

Monitor a MobiLink server farm, by adding the resource to the Monitor.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

All of the servers in the MobiLink server farm must already be added as resources to the Monitor before the MobiLink server farm can be added.

Context

You can add a resource one at a time or you can add multiple resources via a CSV file.

Procedure

1. Log in to the Monitor.
2. Click *Administration*.
3. Click *Resources*.
4. Click *Add*.
5. Follow the instructions in the *Add Resource* window to add a resource to monitor a MobiLink server farm.

   The Monitor collects metrics about the MobiLink server farm, as well as each server that is part of the farm. When you add a MobiLink server farm as a resource, you must specify the servers that are part of the farm.
6. Click *Create*.

7. Click *Close*.
8. Click *Close*.

Results

The resource is added and monitoring of the resource starts automatically. The resource appears in the *Overview* dashboard *Resource List*.

Next Steps

Optionally, you can add a dashboard for the resource. By default, when a resource is added a dashboard is not.

Related Information

SQL Anywhere Scale-out Topology Widget [page 1281]
MobiLink Users in a Synchronization System
Adding Multiple Resources with a CSV File [page 1295]
Managing Widgets [page 1280]
Managing Dashboards [page 1282]
Starting the Monitor [page 1271]
Host Connection Parameter [page 86]
List of Database Objects Installed by the Monitor [page 1351]

1.8.2.1.10.4 Adding a Resource to Monitor the Availability of a Web Server, Proxy Server, or Host Computer

Create a web service resource to monitor the availability of a web server, proxy server, or host computer, and receive notifications when it is unavailable.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were

previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

Procedure

1. In the Monitor, click *Administration*.
2. Click *Resources*, and then click *Add*.
3. Click *Web Service - web server, proxy, or host*, and then follow the instructions in the *Add Resource* window.
4. Click *Create*.
5. Click *Close* twice.

Results

The resource is added and monitoring of the resource starts. The resource appears in the *Overview* dashboard *Resource List*.

Example

Create a web service resource to monitor the availability of a different instance of the Monitor.

1.8.2.1.10.5 Adding Multiple Resources with a CSV File

Add multiple resources to the Monitor by creating a CSV (comma-separated values) file, and then importing the file.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

Context

You cannot use a CSV file to add a primary database in a database mirroring system as a resource; use the *Administration* window instead.

Procedure

1. Create a CSV file.

   Each line in the CSV file contains information about a single resource. Each comma-separated term within a line describes an attribute of the resource. The order of the terms is important.

   For terms that are optional, any preceding terms (including other optional terms) must be specified. For example, to specify the database name, you must also specify the database server name and port number. To specify the encryption type, you must also specify the connection type. To specify the Monitoring URL suffix, you must also specify the port number.

   CSV file format for database resources

Position in the line	Term	Description
1	Resource type	Type `sa` to indicate that the resource is a database. This term is required.
2	Resource name	Specify the resource name that the database will have in the Monitor. This term is required.
3	Username	Specify the user name used to connect to the database. As of version 16, to add a database, you must specify a user name for a user in the database that has the SYS_SAMONITOR_ADMIN_ROLE role. To add an earlier database, you must specify the user name for a user in the database that has DBA authority.
4	Password	Specify the user password that is used to connect to the database. This term is required.
5	Host	Specify the hostname or IP address of the computer on which the database server is running. You can use the name localhost to represent the current computer. This term is required.

Position in the line	Term	Description
6	Port	Specify the port number on which the database server is running. The default port number is 2638. This term is optional. The default value is 0.
7	Server	Specify the name of the database server that the database is connected to. This term is optional.
8	Database	Specify the name of the database. This term is optional.
9	Connection Parameters	Specify additional connection parameters to use when connecting to the database. List the connection parameters as a semicolon-delimited list of option=value pairs. The default value is "".This term is optional.

The following is an example of a CSV file that contains two database resources.

```
sa,demo,DBA,sql,localhost,2638,demo17
sa,demo3,DBA,sql,localhost,49152,demo17,demo
```

CSV file format for MobiLink resources

Position in the line	Term name	Description
1	Resource type	Type **ml** to indicate that the resource is a MobiLink server. This term is required.
2	Resource name	Specify the resource name as it will appear in the Monitor. This term is required.
3	Username	Specify the username used to connect to the resource. This term is required.
4	Password	Specify the password used to connect to the resource. This term is required.
5	Host (MobiLink server)	Specify the hostname or IP address of the computer on which the server is running. You can use the name localhost to represent the current computer. This term is required.

Position in the line	Term name	Description
6	Port (MobiLink server)	Specify the port number on which the MobiLink server is running. The default port number for MobiLink is 2439. This term is optional. The default value is 0.
7	Connection type (MobiLink server)	Specify the method to use to connect to the resource. Type one of the following: • *tcpip* This is the default. • *http* • *https* • *tls* This term is optional.
8	Encryption type (MobiLink server)	Specify the method used to encrypt the connection. Type one of the following: **N** No encryption. This is the default. **R** RSA encryption. **F** FIPS-certified RSA encryption. This term is optional.
9	Connection parameters (MobiLink server)	Specify additional connection parameters to use when connecting to the database. List the connection parameters as a semicolon-delimited list of option=value pairs. This term is optional.

The following is an example of an import file that contains two MobiLink server resources.

```
ml,demo2,DBA,sql,localhost
ml,demo4,DBA,sql,localhost,0,tcpip,N
```

CSV file format for MobiLink server farm resources

Position in the line	Term name	Description
1	Resource type	Type **mf** to indicate that the resource is a MobiLink server farm. This term is required.
2	Resource name	Specify the resource name as it will appear in the Monitor. This term is required.
3	MobiLink server name	Specify the MobiLink server resources that are to be included in the MobiLink server farm. List the MobiLink server resources as a comma-delimited list. This term is required.

The following is an example of an import file that contains two MobiLink server resources and a MobiLink server farm resource.

```
ml,demo2,DBA,sql,localhost
ml,demo4,DBA,sql,localhost,0,tcpip,N
mf,collection_demo,demo2,demo4
```

CSV file format for a web service resource

Position in the line	Term Name	Description
1	Resource type	Type **ws** to indicate that the resource is a web service. This term is required.
2	Resource name	Specify the resource name as it will appear in the Monitor. This term is required.
3	Service type	Specify the type of service to be monitored. Acceptable values are: **W** Web server **P** Proxy **H** Host
4	Use Proxy	Specify **True** to use a proxy when attempting to connect to the web service. The default is **False**. This term is optional.

Position in the line	Term Name	Description
5	Proxy Host	Specify the host of the proxy that will be used if Use Proxy is set to `True`. If left blank, the default proxy host used by web browsers is used. This term is optional.
6	Proxy Port	Specify the port of the proxy that will be used if Use Proxy is set to `True`. If left blank, the default proxy port used by web browsers is used. This term is optional.
7	Description	Specify a description for the web service. This term is optional.

2. Log in to the Monitor.
3. Click *Administration*.
4. Click *Resources*.
5. Click *Import*.
6. Locate the CSV file and click *Open*.

   The resources from the CSV file are added to the Monitor and monitoring starts.
7. Click *Close*.
8. Click *Close*.

Results

The imported resource is added to the *Resource List* in the *Overview* dashboard.

Related Information

Adding a Database Resource [page 1289]
Adding a MobiLink Server Resource [page 1291]

1.8.2.1.10.6 Start Monitoring Resources

There are several conditions under which monitoring a resource starts.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were

previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

- Automatically when the Monitor starts. When you start the Monitor, by default, it connects to the resources and collects metrics for *all* resources in the Monitor.
- Automatically when a Monitor administrator adds a resource. After a resource is added, the Monitor attempts to connect to the resource and starts monitoring it.
- Automatically at the end of a blackout period. The Monitor automatically attempts to connect to the resource and resume monitoring.
- When a Monitor administrator opens the *Administration* window, clicks *Resources*, selects a resource from the list, and clicks *Start*.

Related Information

Adding a Database Resource [page 1289]
Adding a MobiLink Server Resource [page 1291]

1.8.2.1.10.7 Stop Monitoring Resources

You stop monitoring resources when you do not want the Monitor to collect metrics.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

For example, you want to stop monitoring when you know that the resource will be unavailable; otherwise, you receive alerts until the resource is available. Except for the default Monitor resource, you can stop monitoring any resource at any time.

When you stop monitoring a resource, the Monitor:

- Stops collecting metrics for the resource.
- Stops issuing alerts for the resource.

There are two ways to stop monitoring a resource:

Schedule a regular, repeating, blackout period

This method is a good choice when the following conditions apply:

- You must repeatedly stop monitoring the resource. For example, you perform regular maintenance at the end of each month.
- You know in advance how long the resource is unavailable. For example, you know that your regular maintenance takes four hours.

- You need monitoring to automatically restart. When a blackout completes, the Monitor attempts to reconnect to the resource and to continue collecting data.

To use this method, you configure blackouts to make the Monitor automatically stop monitoring at specified times.

Manually stop monitoring

This method is a good choice when the following conditions are met:

- You need to stop monitoring for infrequent or one-time tasks. For example, you need to stop monitoring because the computer that the resource is running on needs to be taken off-line for special maintenance.
- You are available to restart the monitoring afterward. When a resource has been stopped manually, the Monitor waits for you to restart monitoring.

To permanently stop monitoring a resource, you can remove it from the Monitor.

In this section:

Configuring Blackouts [page 1302]
Use the Monitor to configure a blackout.

Stopping Monitoring a Resource [page 1304]
Stop monitoring a resource.

Related Information

Removing Resources [page 1306]

1.8.2.1.10.7.1 Configuring Blackouts

Use the Monitor to configure a blackout.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

Context

Blackouts are times when you do not want the Monitor to collect metrics, and occur in the local time of the resource.

Completed blackouts prompt the Monitor to reconnect to the resources to continue collecting data. When a blackout completes, the Monitor reconnects to the resources to continue collecting data.

Procedure

1. Log in to the Monitor.
2. Click *Tools* > *Administration*.
3. Click *Resources*.
4. Click the resource, and then click *Configure*.
5. Click *Blackouts*.
6. Click *New*.
7. In the *New Blackout Period* window, specify the date and time (24 hour clock) for the blackout.

   The time is local to the computer where the resource database or MobiLink server resides.
8. Click *Save*.
9. Click *Save*.
10. Click *Close*.
11. Click *Close*.

Results

The blackout is configured.

Related Information

Understanding How Time Is Displayed [page 1284]
Adding a Database Resource [page 1289]
Adding a MobiLink Server Resource [page 1291]
Stopping Monitoring a Resource [page 1304]

1.8.2.1.10.7.2 Stopping Monitoring a Resource

Stop monitoring a resource.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

Procedure

1. Log in to the Monitor.
2. Click *Tools* > *Administration*.
3. Click *Resources*.
4. Click the resource, and then click *Stop*.

   > **i** Note
   >
   > When you stop monitoring a MobiLink server farm, monitoring stops for each MobiLink server that is part of the farm.
5. Click *Close*.

Results

The monitoring of the specified resource is stopped.

Related Information

Adding a Database Resource [page 1289]
Adding a MobiLink Server Resource [page 1291]
Configuring Blackouts [page 1302]

1.8.2.1.10.8 Repairing a Database Resource Monitored by the Monitor

Repair a database resource to allow the Monitor to resume monitoring it.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

As of version 16, each time you repair a resource, you must specify a user ID and password for a user in the database that has the SYS_SAMONITOR_ADMIN_ROLE role. Each time you repair earlier database, you must specify the user ID and password for a user in the database that has DBA authority.

Context

Repairing a resource reinstalls the database objects for the resource. You can repair only database resources. You cannot repair the default resource for the Monitor (named *SQL Anywhere Monitor*).

Procedure

1. Log in to the Monitor.
2. Click *Administration*.
3. Click *Resources*.
4. Click the database resource to repair.
5. If the resource is currently being monitored, click *Stop*.
6. Click *Repair*.
7. When prompted, type the user ID and password for the database. The user credentials are used to connect to the database, and then they are removed from the Monitor.
8. Click *Repair*.
9. Click *OK*.
10. Click *Start* to restart monitoring the resource.
11. Click *Close*.

Results

The resource is repaired, and monitoring options are left unchanged.

Related Information

List of Database Objects Installed by the Monitor [page 1351]

1.8.2.1.10.9 Removing Resources

Remove a resource to cause the Monitor to permanently stop monitoring the resource and discard the metrics collected for the resource.

Prerequisites

You must be a Monitor administrator.

Context

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You should only remove resources when you are certain that you do not need to monitor them, such as when a resource is no longer being used. You cannot delete the *SQL Anywhere Monitor* resource.

When you remove a database resource, the Monitor does not delete the monitoring objects installed in the database.

Procedure

1. Log in to the Monitor.
2. Click *Administration*.

3. Click *Resources*.
4. Click the resource.
5. Click *Remove*.
6. Click *Yes*.

> **i Note**
>
> When you remove a MobiLink server farm, the Monitor prompts you to remove the MobiLink servers that are part of the farm. Click *Yes* to remove the identified MobiLink servers, or click *No* remove only the MobiLink server farm and not the identified MobiLink servers.

7. Click *Close*.

Results

The resource is removed.

Related Information

Stop Monitoring Resources [page 1301]
Deleting Monitoring Objects from the Resource Database [page 1352]

1.8.2.1.11 Metrics

The Monitor collects and stores many metrics from databases, MobiLink servers, and MobiLink server farms.

> **i Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

This includes but is not limited to:

- Whether the resource is running.
- Whether the computer that the resource is running on is running properly and is connected to the network.
- Whether the resource is listening and processing requests.
- Whether there are any obvious problems such as long running queries or blocked users.
- The number of synchronizations that the MobiLink server performs over a period of time.
- The average time it takes for the backend server to process an HTTP request.

In this section:

1.8.2.1.11.1 Defining Custom Metrics for a SQL Anywhere Database Resource

Customize a metric to collect data from a resource and receive alerts for that metric in the Monitor.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must have either the CREATE PROCEDURE or CREATE ANY OBJECT to create the function in the resource database.

You must have EXECUTE to grant the EXECUTE privilege to the Monitor user, sa_monitor_user.

Context

Custom metrics allow users to define the data collected by the Monitor for a database resource. Alerts based on the user-specified alert criteria appear in the Alerts List widget.

The SQL Anywhere Monitor records custom metric data by calling the value function in the SQL Anywhere resource every collection period.

Procedure

1. In the resource database:
   a. Create a user-defined function.
   b. Grant the sa_monitor_user EXECUTE privilege for this function.
2. In the Monitor:
   a. In the *Tools* pane, click *Administration*. Select the resource and click *Configure*.
   b. Click the *Custom Metrics* tab.
   c. Click *New*.
   d. Specify the settings for the user-defined function.
   e. Click *OK*.
   f. Click *Save*.
   g. Click *Close*.

Results

The custom metric is created. To specify the alert threshold for the metric, click the *Alert Threshold* tab, and then click *Custom Metrics*. Follow the instructions.

Example

1. A database user wants to:
   - Monitor the numbers of orders in a table.
   - Raise an alert when the total number of orders exceeds 100000.
   1. In the resource database, the user creates the following user-defined function:

```
CREATE FUNCTION "dba"."get_order_count"()
    RETURNS INT
    NOT DETERMINISTIC
    BEGIN
        DECLARE "currentCount" INT;
        SELECT COUNT(*)   INTO currentCount FROM SalesOrders;
        RETURN "currentCount";
END;
```

   2. In the Monitor, the user configures a custom metric with the following settings:

Table 1: Configuring a custom metric

Setting	Value
Name	**OrderCount**
Display units	**Orders**
Minimum	**0**
Maximum	
Data type	**Integer**
Value function owner	**dba**
Value function name	**get_order_count**

3. The user configures an alert with a threshold value of 100000.
   A widget is configured in the Monitor to display a graph for the OrderCount metric.

2. A different user is interested in:
   - Receiving alerts when a transaction takes longer than 5 seconds to complete.
   1. In the resource database, the user creates the following table and event:

```
CREATE TABLE "dba"."order_transaction_time" (
    "pkey" INT PRIMARY KEY DEFAULT AUTOINCREMENT,
    "transaction_time" TIMESTAMP NOT NULL,
    "number_of_seconds" INTEGER NOT NULL
);
CREATE EVENT
"DBA"."truncateData" SCHEDULE "truncateSchedule" START TIME '23:59' EVERY
24 HOURS
HANDLER
BEGIN
    // delete data older than 2 weeks
    DELETE FROM order_transaction_time
    WHERE DATEDIFF(second,transaction_time, current time) > 1209600;
END;
```

   When a transaction completes, an entry is added to the order_transaction_ time table to record the time when the transaction completed and the length of time it took the transaction to run.
   2. The user creates the user-defined function average_order_time_five_min_average and grants EXECUTE privilege on this function to the Monitor user, sa_monitor_user. This function returns the average transaction time over the last 5 minutes.

```
CREATE FUNCTION "dba"."average_order_time_five_min_average"()
    RETURNS INTEGER
    NOT DETERMINISTIC
    BEGIN
        DECLARE "max_order_time" INTEGER;
        // return average transaction time over the last 5 minutes
        SELECT IF AVG(number_of_seconds) IS NOT NULL
            THEN CAST(AVG(number_of_seconds) AS INTEGER )
        ELSE
            0
        ENDIF
        INTO max_order_time
        FROM "DBA"."order_transaction_time"
        WHERE DATEDIFF(second,transaction_time,current time) < 300;
        RETURN max_order_time;
END
```

```
GRANT EXECUTE ON max_order_time to sa_monitor_user.
```

3. In the Monitor, the user creates an average order time metric.

   Table 2: Configuring a custom metric

Setting	Value
Name	`Average order time`
Data type	`Integer`
Display units	`Seconds`
Minimum	`0`
Maximum	
Value function owner	`dba`
Value function name	`order_time_five_min_average`

4. The user configures an alert with a threshold value of 5.
   An alert is triggered when the average transaction time exceeds five seconds.

Related Information

Specifying Alert Thresholds [page 1345]
Creating a User-defined Function

1.8.2.1.11.2 List of Metrics for Database Resources

The Monitor collects and stores many metrics from databases.

> **i Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Metric name	Description
Active HTTP Requests	Returns the number of HTTP connections that are actively processing an HTTP request. An HTTP connection that has sent its response is not included.
Active HTTPS Requests	Returns the number of secure HTTPS connections that are actively processing an HTTPS request. An HTTPS connection that has sent its response is not included.

Metric name	Description
Arbiter State	Returns one of the following values: **Connected** The arbiter server is connected to the primary server. **Disconnected** The arbiter server is not connected to the primary server. **0** There is no arbiter server in this configuration.
Cache Dirty	Returns the number of cache pages that are dirty (needing a write).
Cache Pinned	Returns the number of pinned cache pages.
Cache Replacements	Returns the number of pages in the cache that have been replaced.
Cache Size	Returns the current cache size, in kilobytes.
Connection Count	Returns the current number of connections to the database.
CPU Usage	Returns the percentage of CPU time used by the database server.
Database Name	Returns name of the database.
Disk Reads	Returns the rate at which data is being read from the disk (in kilobytes per second). This value is calculated based on the DiskRead property.
Disk Writes	Returns the rate at which data is being written to the disk (in kilobytes per second). This value is calculated based on the DiskWrite property.
File Size	Returns the size of the main database file.
Free Disk Space	Returns the amount of free space on your disk.
Host	Returns the name of the computer running the database server. Typically, this is the computer's host name.
HTTP Connections	Returns the number of HTTP connections that are currently open within the database server. They may be actively processing a request or waiting in a queue of long lived (keep-alive) connections.
HTTP Sessions	Returns the number of active and dormant HTTP sessions within the database server.

Metric name	Description
HTTPS Connections	Returns the number of HTTPS connections that are currently open within the database server. They may be actively processing a request or waiting in a queue of long lived (keep-alive) connections.
Last Checked Time	Returns the last time the Monitor collected data for the resource.
License Type	Returns the license type. Can be networked seat (per-seat) or CPU-based.
Licensed Company	Returns the name of the licensed company.
Licensed Seats	Returns the number of licensed seats or processors.
Licensed User	Returns the name of the licensed user.
Long Running Queries	Returns queries that exceed the specified long running query threshold.
Main Heap Pages	Returns the number of pages used for global server data structures.
Max Cache Size	Returns the maximum allowed cache size.
Mirror Mode	Returns *Mirroring Is Not Enabled On This Database* if database mirroring is not in use. If mirroring is enabled, shows *Synchronous* if the synchronization mode specified with the SET MIRROR OPTION statement is synchronous, and *Asynchronous* otherwise.
Mirror State	Returns one of the following values: **Synchronizing** The mirror server is not connected or has not yet read all the primary server's log pages. This value is also returned if the synchronization mode is asynchronous. **Synchronized** The mirror server is connected and has all changes that have been committed on the primary server.
Operating System	Returns the operating system on which the software is running.
Operating System Version	Returns the operating system on which the software is running, including build numbers and service packs.

Metric name	Description
Partner State	Returns one of the following values: **Connected** The mirror server is connected to the primary server. **Disconnected** The mirror server is not connected to the primary server.
Peak Cache Size	Returns the largest value the cache has reached in the current session, in kilobytes.
Processor Architecture	Returns a string that identifies the processor type.
Queries Processed	Returns the rate (queries/second) at which queries are processed.
Seat Count	Returns the number of unique client network addresses connected to a network database server.
Server Language	Returns the locale language, which is the language that is expected to be used by users of the database server.
Server Name	Returns the name of the database server for the current connection.
Server Time Offset	Returns the time difference between the time on the computer that the Monitor is running and the time on the computer that you are using to view the Monitor data.
Server Type	Returns the type (personal or network) of database server being monitored.
Server Version	Returns the version of the software being run.
Start Time	Returns the time when the database started.
Total Disk Space	Returns the size of your disk.
Unavailable Since	Returns the time the resource became unavailable.
Unscheduled Requests	Returns the number of unscheduled requests.
Unsubmitted Error Reports	Returns the number of unsubmitted error reports for the database. An error report is submitted when the software crashes.

1.8.2.1.11.3 List of Metrics for MobiLink Server Resources

The Monitor collects and stores many metrics from MobiLink server resources.

> **i Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Metric name	Description
Authenticating Synchronizations	Shows the number of requests in the server that are currently in the authenticate user phase.
Cached Remote Tasks Requests	Shows the number of cached remote tasks requests.
Command Processor Stage Length	Shows the length of the queue for synchronization work.
Commit Rate	Shows the rate at which commits occur.
Commits	Shows total number of commits.
Completed Synchronization Rate	Shows the rate of successfully completed synchronizations for the server.
Connected Clients	Shows the number of connected clients.
Connections Closed Rate	Shows the rate that TCP connections are closed.
Connections Opened Rate	Shows the rate that TCP connections are opened.
Connections Rejected Rate	Shows the rate that TCP connections are rejected by the server.
Consolidated Database Type	Shows the type of consolidated database. For example, *SQL Anywhere*.
Consolidated Database Version	Shows the version of the consolidated database.
CPU Total Time	Shows the amount of CPU time used by the MobiLink server in microseconds.
CPU Usage	Shows the percentage of CPU time used by the MobiLink server.
CPUs	Shows the number of CPUs in use.
Current TCP Connections	Shows the number of TCP connections currently in use by this MobiLink server.

Metric name	Description
Database Connections In Use	Shows the number of database connections in use.
Database Worker Thread Count	Shows the number of database worker threads.
Downloading Synchronizations	Shows the current number of synchronizations that are in the Prepare for Download, Fetch Download, Wait for Download Ack, or End Sync phases.
Driver Name	Shows the name of the driver for the consolidated database.
Driver Version	Shows the version of the driver for the consolidated database.
Errors	Shows the total number of errors.
Failed Synchronization Rate	Shows the rate of failed synchronizations for the server, expressed in synchronizations per second.
Failed Synchronizations	Shows total number of failed synchronizations.
File Transfers	Shows the number of mlfiletransfers currently connected.
Finished Synchronization Rate	Shows the rate at which synchronizations complete.
Free Disk Space For MobiLink Cache	Shows the disk space available on the temp disk for MobiLink cache in bytes.
Heartbeat Stage Length	Shows the length of the queue for periodic, non-synchronization work.
Host	Shows the name of the computer running the MobiLink server. Typically, this is the computer's host name.
Is Consolidated Available?	Show *True* if the MobiLink server can connect to the consolidated database.
Is Consolidated Database Clustered?	Shows *True* if the MobiLink server is using a clustered consolidated database.
Is Primary Server Known?	Shows *Yes* if the primary server is known by the server. Shows *No* if the primary server is unknown or if the server does not care what the primary server is.
Is Primary Server?	Shows *Yes* when the server is the primary server.
Last Checked Time	Shows the last time the Monitor collected data for the resource.
Licensed Company	Shows the name of the licensed company.
Licensed User	Shows the name of the licensed user.
Listeners	Shows the number of listeners currently connected.

Metric name	Description
Max Cache Size	Shows the maximum cache size used by the MobiLink server.
Max Clients	Shows the maximum number of clients.
Max Database Connections	Shows the maximum number of database connections, as set by the -cn option or the -w option for mlsrv17.
Max Synchronization Time	Shows the maximum time it took for a synchronization to occur. The age of the oldest synchronization in microseconds.
Max TCP Connections	Shows the maximum number of TCP connections, as set by the -nc option for mlsrv17.
Max Uploads Allowed	Shows the maximum number of concurrent uploads to the database.
Max Wait For Connection	Shows the longest length of time an active synchronization has been waiting for the database.
Memory Used	Shows the bytes of RAM in use. Shows the Tracked Memory value when the server runs on a non-Windows platform.
Monitors	Shows the number of Monitors and MobiLink Profilers currently connected.
Notifier Stage Length	Shows the length of the notifier work queue.
Operating System	Shows the operating system on which the software is running.
Pages In Stream Stack	Shows the number of pages held by the network streams.
Pages Locked	Shows the number of cache pages loaded into memory.
Pages Swapped In	Shows the total number of pages ever read from disk.
Pages Swapped In Rate	Shows rate at which the pages are read from disk.
Pages Swapped Out	Shows the total number of pages ever swapped to disk.
Pages Swapped Out Rate	shows the rate at which the pages are swapped to disk.
Pages Used	The number of cache pages used. This includes pages swapped to disk so it may be larger than the cache size.
Percent of Clients Connected	Shows the percentage of clients that are currently connected. This metric is derived from dividing the *Number Of Connected Clients* value by the *Maximum Clients* value and multiplying the result by 100.

Metric name	Description
Percent of Connections In Use	Shows the percentage of connections that are currently in use.
Percent of Pages in Stream Stack	Shows the percentage of pages that are held by the network streams.
Percent of Pages Locked	Shows the percentage of pages that are locked.
Percent of Pages Used	Shows the percentage of pages that are in use. This includes pages swapped to disk so it may be larger than the cache size.
Pings	Shows the number of pinging clients currently connected.
Processor Architecture	Shows a string that identifies the processor type.
Raw TCP Stage Length	Shows the length of the network work queue.
Requests	Shows the number of valid requests being serviced.
Requests in Apply Upload Phase	Shows the number of synchronizations currently in the apply upload phase.
Requests in Authenticate User Phase	Shows the number of synchronizations currently in the authenticate user phase.
Requests in Begin Synchronization Phase	Shows the number of synchronizations currently in the begin synchronization phase.
Requests in Connect for Download Ack Phase	Shows the number of synchronizations currently in the connect for download ack phase.
Requests in Connect Phase	Shows the number of synchronizations currently in the connect phase.
Requests in End Synchronization Phase	Shows the number of synchronizations currently in the end synchronization phase.
Requests in Fetch Download Phase	Shows the number of synchronizations currently in the fetch download phase.
Requests in Get DB Worker For Download Ack Phase	Shows the number of synchronizations currently in the get DB worker for ack phase.
Requests in Get DB Worker Phase	Shows the number of synchronizations currently in the get DB worker phase.
Requests in Non-Blocking Download Ack Phase	Shows the number of synchronizations currently in the non-blocking download ack phase.

Metric name	Description
Requests in Prepare For Download Phase	Shows the number of synchronizations currently in the prepare for download phase.
Requests in Receive Upload Phase	Shows the number of synchronizations currently in the receive upload phase.
Requests in Send Download Phase	Shows the number of synchronizations currently in the send download phase.
Requests in Synchronization Request Phase	Shows the number of synchronizations currently in the synchronization request phase.
Requests in Wait For Download Ack Phase	Shows the number of synchronizations currently in the wait for download ack phase.
Rollback Rate	Shows the rate at which rollbacks occur.
Rollbacks	Shows the total number of rollbacks.
Rows Downloaded	Shows the total number of rows sent to remotes.
Rows Downloaded Rate	Shows rate at which rows are sent to remotes.
Rows Uploaded	Shows the total number of rows received from remotes.
Rows Uploaded Rate	Shows the rate at which rows are received from remotes.
RTNotifier Light Weight Poll Hit Rate	Shows the rate of the number of lightweight polls from remote tasks agents that found at least one remote tasks request.
RTNotifier Light Weight Poll Hits	Shows the number of lightweight polls from remote task agents that found at least one remote task request.
RTNotifier Light Weight Poll Miss Rate	Shows the rate of the number of lightweight polls from remote tasks agents that found no remote tasks requests.
RTNotifier Light Weight Poll Misses	Shows the number of lightweight polls from remote task agents that found no remote tasks.
RTNotifier Light Weight Poll Rate	Shows the rate of the number of lightweight polls from remote task agents.
RTNotifier Light Weight Polls	Shows the number of lightweight polls from remote task agents.
Server Cache Size	Shows the size of the server cache.

Metric name	Description
Server Name	Shows the name of the MobiLink server as specified by the -zs option for the connected server. The default value is *<default>*.
Server Time Offset	Shows the time difference between the time on the computer that the Monitor is running and the time on the computer that you are using to view the Monitor data.
Server Tracked Memory Usage	Shows the amount of memory allocated by the server. Use this metric for non-Windows systems where the MEMORY_USED metric is unavailable. On Microsoft Windows systems, use the MEMORY_USED metric for increased accuracy.
Start Time	Shows the time when the MobiLink server started.
Started Synchronization Rate	Shows the rate at which synchronizations are started.
Stream Stage Length	Shows the length of the high level network processing queue.
Successful Synchronizations	Shows the total number of successful synchronizations.
Synchronization Error Rate	Shows the rate of synchronization errors.
Synchronization Warning Rate	Shows the rate of synchronization warnings for the server.
Synchronizations	Shows the number of data synchronizations currently connected. See NUM_CONNECTED_SYNCS.
Synchronizations Blocked	Shows the number of synchronizations that are currently waiting for a database connection.
Synchronizations Finished	Shows the total number of synchronizations that have finished.
Synchronizations Started	Shows the total number of synchronizations that have started.
TCP Bytes Read	Shows the total number of bytes ever read.
TCP_Bytes Read Rate	Shows rate at which bytes are read from the network.
TCP Bytes Written	Shows the total number of bytes ever written.
TCP_Bytes Written Rate	Shows rate at which bytes are written to the network.
TCP Connections Closed	Shows total number of connections ever closed.
TCP Connections Opened	Shows the total number of connections ever opened.

Metric name	Description
TCP Connections Rejected	Shows the total number of connections ever rejected.
Timed Work Stage Length	Shows the length of the dynamic caching work queue.
Unavailable Since	Shows the time the resource became unavailable.
Unknown Connected Clients	Shows the number of connected clients whose origins are unknown.
Unsubmitted Error Reports	Shows the number of unsubmitted error reports for the server. An error report is submitted when the software crashes.
Upload Connections In Use	Shows the number of upload connections currently in use.
Uploading Synchronization	Shows the number of synchronizations that are in the upload phase.
Version	Shows the version of the software being run.
VM Memory Usage	Shows the amount of memory used by any attached VMs.
Waiting For Database Connection	Number of requests that are waiting for a database connection.
Warnings	Shows the total number of warnings.

1.8.2.1.11.4 List of Metrics for MobiLink Server Farm Resources

The Monitor collects and stores many metrics from MobiLink server farm resources.

> **i Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Metric name	Scope	Description
Authenticating Synchronizations	Average	Shows the number of requests in the server that are currently in the authenticate user phase.
Bytes Read Rate	Average	Shows the rate at which bytes are read from the network.

Metric name	Scope	Description
Bytes Written Rate	Average	Shows the rate at which bytes are written to the network.
Cached Remote Task Requests	Average	Shows the number of cached remote tasks requests.
Command Processor Stage Length	Average	Shows the length of the queue for synchronization work.
Commit Rate	Average	Shows the rate at which commits occur.
Commits	Average	Shows total number of commits.
Completed Synchronization Rate	Total	Shows the rate of successfully completed synchronizations for the server.
Connected Clients	Average	Shows the number of connected clients.
Connections Closed Rate	Average	Shows the rate that TCP connections are closed.
Connections Opened Rate	Average	Shows the rate that TCP connections are opened.
Connections Rejected Rate	Average	Shows the rate that connections are rejected by the server.
CPU Total Time	Total	Shows the amount of CPU time used by the MobiLink server in microseconds.
CPU Usage	Average	Shows the average amount of CPU used by the MobiLink server.
Current TCP Connections	Total	Shows the number of TCP connections currently in use by this MobiLink server.
Database Connections In Use	Total	Shows the number of database connections in use.
Downloading Synchronizations	Average	Shows the current number of synchronizations that are in the Prepare for Download, Fetch Download, Wait for Download Ack, or End Sync phases.
Errors	Average	Shows the average number of errors.
Failed Synchronization Rate	Total	Shows the rate of failed synchronizations for the server.

Metric name	Scope	Description
Failed Synchronizations	Average	Shows total number of failed synchronizations.
File Transfers	Total	Shows the number of mlfiletransfers currently connected.
Finished Synchronization Rate	Total	Shows the rate at which synchronizations complete.
Free Disk Space For MobiLink Cache	Average	Shows the disk space available on the temp disk for MobiLink cache in bytes.
Heartbeat Stage Length	Average	Shows the length of the queue for periodic, non-synchronization work.
Listeners	Total	Shows the number of listeners currently connected.
Max Synchronization Time	Maximum	Shows the maximum time it took for a synchronization to occur. The age of the oldest synchronization in microseconds.
Max Wait For Connection	Maximum	Shows the longest length of time an active synchronization has been waiting for the database.
Memory Used	Average	Shows the bytes of RAM in use. Shows the Tracked Memory value when the server runs on a non-Windows platform.
Monitors	Total	Shows the number of Monitors and MobiLink Profilers currently connected.
Notifier Stage Length	Average	Shows the length of the notifier work queue.
Pages in Stream Stack	Average	Shows the number of pages held by the network streams.
Pages Locked	Average	Shows the number of cache pages loaded into memory.
Pages Swapped In	Average	Shows the total number of pages ever read from disk.
Pages Swapped In Rate	Average	Shows rate at which the pages are read from disk.

Metric name	Scope	Description
Pages Swapped Out	Average	Shows the total number of pages ever swapped to disk.
Pages Swapped Out Rate	Average	Shows the rate at which the pages are swapped to disk.
Pages Used	Average	The number of cache pages used. This includes pages swapped to disk, so it may be larger than the cache size.
Percent of Clients Connected	Average	Shows the percentage of clients that are currently connected. This metric is derived from dividing the *Number Of Connected Clients* value by the *Maximum Clients* value and multiplying the result by 100.
Percent of Connections In Use	Average	Shows the percentage of connections that are currently in use.
Percent of Pages in Stream Stack	Average	Shows the percentage of pages that are held by the network streams.
Percent of Pages Locked	Average	Shows the percentage of pages that are locked.
Percent of Pages Used	Average	Shows the percentage of pages that are in use. This includes pages swapped to disk so it may be larger than the cache size. See PAGES_USED.
Pings	Total	Shows the number of pinging clients currently connected.
Raw TCP Stage Length	Average	Shows the length of the network work queue.
Requests	Total	Shows the number of valid requests being serviced.
Requests in Apply Upload Phase	Average	Shows the number of synchronizations currently in the apply upload phase.
Requests in Authenticate User Phase	Average	Shows the number of synchronizations currently in the authenticate user phase.

Metric name	Scope	Description
Requests in Begin Synchronization Phase	Average	Shows the number of synchronizations currently in the begin synchronization phase.
Requests in Connect for Download Ack Phase	Average	Shows the number of synchronizations currently in the connect for download ack phase.
Requests in Connect Phase	Average	Shows the number of synchronizations currently in the connect phase.
Requests in End Synchronization Phase	Average	Shows the number of synchronizations currently in the end synchronization phase.
Requests in Fetch Download Phase	Average	Shows the number of synchronizations currently in the fetch download phase.
Requests in Get DB Worker For Download Ack Phase	Average	Shows the number of synchronizations currently in the get DB worker for ack phase.
Requests in Get DB Worker Phase	Average	Shows the number of synchronizations currently in the get DB worker phase.
Requests in Non-Blocking Download Ack Phase	Average	Shows the number of synchronizations currently in the non-blocking download ack phase.
Requests in Prepare For Download Phase	Average	Shows the number of synchronizations currently in the prepare for download phase.
Requests in Receive Upload Phase	Average	Shows the number of synchronizations currently in the receive upload phase.
Requests in Send Download Phase	Average	Shows the number of synchronizations currently in the send download phase.
Requests in Synchronization Request Phase	Average	Shows the number of synchronizations currently in the synchronization request phase.
Requests in Wait For Download Ack Phase	Average	Shows the number of synchronizations currently in the wait for download ack phase.
Rollback Rate	Average	Shows the total number of rollbacks.

Metric name	Scope	Description
Rollbacks	Average	Shows the total number of rollbacks.
Rows Downloaded	Average	Shows the total number of rows sent to remotes.
Rows Downloaded Rate	Average	Shows the total number of rows sent to remotes.
Rows Uploaded	Average	Shows the total number of rows received from remotes.
Rows Uploaded Rate	Average	Shows the total number of rows received from remotes.
RTNotifier Light Weight Poll Hit Rate	Average	Shows the rate of the number of lightweight polls from remote tasks agents that found at least one remote tasks request.
RTNotifier Light Weight Poll Hits	Average	Shows the number of lightweight polls from remote task agents that found at least one remote task request.
RTNotifier Light Weight Poll Miss Rate	Average	Shows the rate of the number of lightweight polls from remote tasks agents that found no remote tasks requests.
RTNotifier Light Weight Poll Misses	Average	Shows the number of lightweight polls from remote task agents that found no remote tasks.
RTNotifier Light Weight Poll Rate	Average	Shows the rate of the number of lightweight polls from remote task agents.
RTNotifier Light Weight Polls	Average	Shows the number of lightweight polls from remote task agents.
Server Cache Size	Average	Shows the size of the server cache.
Server Tracked Memory Usage	Average	Shows the amount of memory allocated by the server. Use this metric for non-Windows systems where the MEMORY_USED metric is unavailable. On Microsoft Windows systems, use the MEMORY_USED metric for increased accuracy.
Started Synchronization Rate	Total	Shows the rate at which synchronizations are started.

Metric name	Scope	Description
Stream Stage Length	Average	Shows the length of the high level network processing queue.
Successful Synchronizations	Average	Shows the total number of successful synchronizations.
Synchronization Error Rate	Total	Shows the rate of synchronization errors.
Synchronization Warning Rate	Total	Shows the rate of synchronization warnings for the server.
Synchronizations	Total	Shows the number of data synchronizations currently connected.
Synchronizations Blocked	Average	Shows the number of synchronizations that are currently waiting for a database connection.
Synchronizations Finished	Total	Shows the number of synchronizations that have been completed.
Synchronizations Started	Total	Shows the total number of synchronizations that have started.
TCP Bytes Read	Average	Shows the total number of bytes ever read.
TCP Bytes Written	Average	Shows the total number of bytes ever written.
TCP Connections Closed	Average	Shows total number of connections ever closed.
TCP Connections Opened	Average	Shows the total number of connections ever opened.
TCP Connections Rejected	Average	Shows the total number of connections ever rejected.
Timed Work Stage Length	Average	Shows the length of the dynamic caching work queue.
Unknown Connected Clients	Average	Shows the number of connected clients whose origins are unknown.
Upload Connections In Use	Average	Shows the number of upload connections currently in use.

Metric name	Scope	Description
Uploading Synchronization	Average	Shows the number of synchronizations that are in the upload phase.
VM Memory Usage	Average	Shows the amount of memory used by any attached VMs.
Waiting for Database Connection	Average	Number of requests that are waiting for a database connection.
Warnings	Average	Shows the average number of warnings.

1.8.2.1.11.5 Setting the Collection Interval for a Resource

Change the rate at which metrics are collected by the Monitor.

Prerequisites

You must be a Monitor administrator.

Context

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Metrics displayed in the Monitor are only as precise as the collection interval, which by default is 30 seconds.

When metrics are refreshed, the collection interval for a resource is different from the refresh rate of the Monitor display.

Procedure

1. Log in to the Monitor.
2. Click *Administration*.
3. Click *Resources*.

4. Click the resource, and then click *Configure*.
5. Click *Collection Interval*.
6. Specify the interval at which metrics are collected (`hours:minutes:seconds`). The collection interval must be at least 10 seconds long.
7. Click *Save*.
8. Enter your user ID and password and click *OK*.
9. Click *Close*.
10. Click *Close*.

Results

The collection interval for the resource is set.

Related Information

Refreshing Metrics [page 1330]

1.8.2.1.11.6 Exporting Metrics

Export metrics that have a graph or table associated with them to an XML file.

Context

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

For example, most of the metrics in the *Key Performance Metrics* widget can be exported. The maximum amount of data that you can export to a file is 25 metrics or 1 million points.

Procedure

1. Log in to the Monitor.

2. Open a dashboard.
3. On a widget that displays the metrics, click the dropdown menu arrow, then click *Export* and follow the instructions in the *Export Metrics* window.

   > **i Note**
   >
   > You cannot export more than 25 metrics or more than one million points. If you do, you receive an error message and are returned to the export screen to select fewer metrics or a smaller time range.

4. Click *Export*.
5. When prompted, specify a file name with the file extension `.xml`.
6. Click *Save*.

Results

An XML file is created containing the specified metrics.

1.8.2.1.11.7 Refreshing Metrics

Change the Monitor's refresh interval, which is independent of the collection interval rate for a resource.

Context

> **i Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

By default, the Monitor display is automatically refreshed every minute.

Procedure

1. Set the refresh rate.
   a. Click *Tools* > *User Settings*.
   b. Change the settings as required. The default is one minute.
   c. Click *Save*.
2. Click *Refresh Data*.

The Monitor retrieves and displays the latest metrics.

Results

The refresh interval is changed.

Next Steps

Press F5 to have the Monitor reload the browser, retrieve the metrics that the Monitor has collected to date, and display the metrics.

Related Information

Metrics [page 1307]

1.8.2.1.12 Monitor Users

Monitor users must log in to the Monitor with a user name and password.

> **i Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

The user name and password for logging in to the Monitor are case sensitive.

The Monitor supports three types of users:

User type	Description
Read-only user	Has read-only access to monitor resources. Read-only users can view metrics, but cannot access the *Administration* window. When you create a user from the log-in screen, the user is a Read-only user. A user name and password are required.

User type	Description
Operator	Has read-only access to monitor resources and can receive alerts. These users can view the metrics, can receive email alerts, and can resolve and delete alerts. However, operators cannot access the *Administration* window. A user name and password are required.
Administrator	Has the same access as an operator, and can also configure resources and add users. Monitor administrators can also access the *Administration* window. The default user, *admin*, is a Monitor administrator. A user name and password are required.

You can check your Monitor user type by logging in to the Monitor, then clicking *Tools* > *User Settings*. You can also change your user settings. Only a Monitor administrator can change a user's type.

Default user for the Monitor

By default, when you first start the Monitor, it has one Monitor administrator user, named *admin*, with the password *admin*. It is recommended that you change the default Monitor administrator password to restrict access to the Monitor.

In this section:

Related Information

Administration Window for the Monitor [page 1285]

1.8.2.1.12.1 Creating Monitor Users

Create any type of Monitor user, including administrators, by using the *Administration* window.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

Procedure

1. Log in to the Monitor.
2. Click *Tools* > *Administration*, and then click *Users*.
3. Click *New* and fill in the information for the new user.

   An email address is only necessary for Monitor administrators and operators who should receive email alerts from the Monitor.

   The specified language sets the language used by the Monitor, including the language used in alerts.
4. Click *Next* and specify the resource dashboards for the user.
5. Click *Save* and *Close*.

Results

A new user is created.

Next Steps

If you created a Monitor administrator or operator, you can associate the user with resources and they can receive alert notifications by email.

Related Information

Associating Monitor Users with Resources [page 1335]
Restricting User Creation to Monitor Administrators [page 1338]
Enabling the Monitor to Send Alert Emails [page 1347]
Editing a Monitor User [page 1336]

1.8.2.1.12.2 Creating Read-only Monitor Users

Create your own read-only user and have read-only access to the monitor.

Context

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

By default, you can create your own read-only user and have read-only access to the Monitor. Monitor administrators can change this default behavior so that only Monitor administrators can create users.

Procedure

1. On the log-in screen, click *Create New User*.

   > **i** Note
   >
   > If the *Create New User* link is not available, then the Monitor administrator has changed the default behavior so that only Monitor administrators can create new users. Contact your Monitor administrator to have them create a new user.

2. Fill in the information for the new user.

   An email address is necessary only to receive email alerts from the Monitor. Only Monitor administrators and operators can receive email alerts. Contact your Monitor administrator to change your user type and enable email alerts.

Results

A new read-only user is created.

Related Information

Associating Monitor Users with Resources [page 1335]
Restricting User Creation to Monitor Administrators [page 1338]
Enabling the Monitor to Send Alert Emails [page 1347]
Editing a Monitor User [page 1336]

1.8.2.1.12.3 Associating Monitor Users with Resources

Associate a Monitor user with a resource when you want the user to receive email alerts about the resource.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

Ensure that the Monitor is set up to send alert notifications by email.

Context

Only Monitor administrators and operators can receive email alerts about resources.

Procedure

1. Log in to the Monitor.
2. Click *Tools* > *Administration*, and then click *Users*.

3. Select a Monitor administrator or operator and ensure that the user has an email address specified in their user account.
4. Click *Email Alert Notification Settings*.
5. Use the *Add* and *Remove* buttons to associate the user with resources. When you are finished, click *OK*.

Results

The user receives email alerts about the specified resources.

Related Information

Enabling the Monitor to Send Alert Emails [page 1347]
Enabling the Monitor to Send Alert Emails [page 1347]

1.8.2.1.12.4 Editing a Monitor User

Edit a Monitor user. All users can change their own user settings, but only a Monitor administrator can change a user's type.

Prerequisites

> **i Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

Procedure

1. Log in to the Monitor.
2. Click *Tools* > *Administration*.
3. Click *Users*.
4. Click the user to edit, and then click *Edit*.

5. Change the settings for the user as required.

   An email address is only necessary for Monitor administrators and operators who should receive email alerts from the Monitor.
6. Click *Save*.
7. Click *Close*.

Results

The Monitor user is edited.

Next Steps

If you created a Monitor administrator or operator, you can associate the user with resources and they can receive alert notifications by email.

Related Information

Associating Monitor Users with Resources [page 1335]
Enabling the Monitor to Send Alert Emails [page 1347]
Creating Monitor Users [page 1333]
Deleting Monitor Users [page 1337]

1.8.2.1.12.5 Deleting Monitor Users

Delete a user from the Monitor and disassociate them from any associated resources.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

Procedure

1. Log in to the Monitor.
2. Click *Tools* > *Administration* and then *Users*.
3. Select the user, and then click *Delete*.
4. Click *Yes* to delete the selected user.
5. Click *Close*.

Results

The user is deleted from the Monitor.

Related Information

Creating Monitor Users [page 1333]
Editing a Monitor User [page 1336]

1.8.2.1.12.6 Restricting User Creation to Monitor Administrators

Disable the default ability for users to create read-only access users for the Monitor.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

Procedure

1. Log in to the Monitor.
2. Click *Tools* > *Administration*.
3. Click *Configuration*.
4. Click *Edit*.
5. Click *Options*.
6. Clear the *Allow Anyone Read-only Access To The SQL Anywhere Monitor* option.
7. Click *Save*.
8. Click *Close*.

Results

Only Monitor administrators can create users.

Related Information

Creating Monitor Users [page 1333]
Editing a Monitor User [page 1336]

1.8.2.1.13 Alerts

An **alert** is a condition or state of interest about a resource that should be brought to a Monitor administrator's or operator's attention.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Alerts are detected by the Monitor based on metrics that are collected. They are not detected at the resource being monitored.

There are predefined alerts for conditions such as low disk space, failed login attempts, and high memory usage. You can change the default threshold values by editing the resource.

When an alert condition is met, the alert is listed in the *Alert List* widget for the specified resource. Also, you can configure the Monitor to send an email to Monitor administrators and operators when an alert occurs.

By default, alerts appear in the *Alert List* widgets and they include information about the cause of the problem, and they provide advice for resolving the problem. In the *Resource List* the resource's status changes to reflect the existence and severity of the alert.

Only the most recent 50 alerts per resource are stored by the Monitor and displayed in the *Alert List* widget; older alerts are deleted.

In this section:

1.8.2.1.13.1 Viewing Alerts

View alerts about resources in the Monitor.

Context

> **Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Only Monitor administrators and operators can resolve and delete alerts.

Only the 100 most recent alerts per resource are stored by the Monitor and displayed in the *Alert List*.

Procedure

1. Click *Alerts* > *Today*.
2. Click an alert in the list, and then click *Details*.

Results

The details of the alert appear in a window box.

Next Steps

To close the details of the alert, click *OK*.

Related Information

Resolving and Deleting Alerts [page 1341]
Enabling the Monitor to Send Alert Emails [page 1347]

1.8.2.1.13.2 Resolving and Deleting Alerts

Mark an alert as resolved or delete an alert after the issue that triggered it has been addressed.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator or operator.

Context

Resolving an alert causes the Monitor to change the alert *Status* to *Resolved By* `user-name`, but leaves the alert in the alert list.

To remove the alert from the list, delete it.

Deleting alerts from the *Alert List* widget does not have any affect on alert emails.

Procedure

1. Log in to the Monitor.
2. Click *Overview*.
3. In the Alert List widget, click an alert and choose one of the following options:

Action	Description
Mark the alert as resolved	In the *Alert List* widget, click an alert in the list and click *Mark Resolved*.
Delete the alert	1. In the *Alert List* widget, click an alert and click *Delete*. 2. Follow the instructions in the window.

Results

- For resolved alerts:
  - The alert is removed from the *Alert List*widget by default.
  - If the alert was the resource's only unresolved alert, then the resource's status in the *Resource List* widget changes to Healthy (no icon is present).
- For deleted alerts:
  - The alert is deleted from the *Alert List* widget.

Related Information

Dashboards and Widgets [page 1277]
Enabling the Monitor to Send Alert Emails [page 1347]
Viewing Alerts [page 1340]

1.8.2.1.13.3 Alert Definitions and Thresholds

As a Monitor administrator, you can configure the thresholds that are used to trigger alerts.

The following list describes the alerts and their default thresholds.

Because each synchronization system has different behaviors and constraints, the defaults may be inappropriate for your environment. You should review each alert threshold and adjust them as necessary.

Alert Thresholds for Database Resources

Alert When CPU Usage Exceeds The Given Threshold For The Given Number of Seconds

Threshold (%)

Issue an alert when the resource's CPU use reaches the specified percentage. The default is 95 percent.

Seconds

The default is 30 seconds.

Alert When Memory Usage Reaches X% Of The Maximum Cache Size

Issue an alert when the memory usage of the resource reaches the specified percentage. The default is 90 percent.

Alert When Free Disk Space Per Dbspace Is Less Than X MB On The Disk

Issue an alert when the free disk space per dbspace is less than the specified amount. The default is 100 MB.

Alert When A Connection Has Been Blocked For Longer Than X Seconds

Issue an alert when a connection has been blocked for longer than the specified time. The default is 10 seconds.

Alert When The Number Of Connections In Use Reaches X % Of The License Limit

Issue an alert when the number of connections in use reaches the specified percentage of the license limit. The default is 85 percent.

Alert When A Query Has Run For Longer Than X Seconds

Issue an alert when a query has run for longer than the specified time. The default is 10 seconds.

Alert When The Number Of Unscheduled Requests Reaches X

Issue an alert when the number of unscheduled requests reaches the specified amount. The default is 5 requests.

Alert When It Has Been More Than The Given Number Of Days Since The Last Server-side Backup

Alert When It Has Been More Than The Given Number Of Days Since The Last Server-side Backup Issue an alert when the number of days since the resource database last had a server-side backup exceeds the specified number of days. The default is 14 days. Note that if the database has never been backed up, this alert check will not be performed.

Alert When The Number Of Disconnected Scale-out Nodes Exceeds The Given Threshold:

Issue an alert when your database is involved in a read-only scale-out or a mirroring system and the system has the specified percentage of disconnected nodes.

Suppress Alerts For The Same Condition That Occur Within X Minutes

This option prevents you from receiving duplicate alerts within a specified time. The default is 30 minutes.

Alert Thresholds for MobiLink Server Resources

Alert When CPU Usage Exceeds The Given Threshold For The Given Number Of Seconds

The *Threshold* default is 100 percent. The *Seconds* default is 300.

Alert When The Percentage Of Cache Pages Used Is Greater Than (%)

The default is 100.

Alert When The Percentage Of Locked Cache Pages Is Greater Than (%)

The default is 80.

Alert When The Number Of Pages Being Swapped In And Out Per Second Exceeds The Given Threshold For The Given Number Of Seconds

The threshold default is 256. The seconds default is 120.

Alert When Longest Active Synchronization Time Is Greater Than (Seconds)

The default is 600.

Alert When The Number Of Failed Synchronizations Exceeds The Given Threshold For The Given Number Of Minutes

The threshold of failed synchronizations default is 20. The minutes default is 60.

Alert When The Number Of Errors Exceeds The Given Threshold For The Given Number Of Minutes

The Threshold (Errors) default is 50. The Minutes default is 60.

Alert When Free Disk Space For MobiLink Cache Is Less Than (MB)

The default is 100.

Alert When The Longest Active Wait For Database Worker Thread Is Greater Than (Seconds)

The default is 300.

Suppress Alerts For The Same Condition That Occur Within Minutes

This option prevents you from receiving duplicate alerts within a specified time. The default is 30 minutes.

Alert Thresholds for Web Service Resources

Alert When a Web Service is Unavailable

Issue an alert when the web service has been unavailable for the specified period of time.

Suppress Alerts For The Same Condition That Occur Within Minutes

This option prevents you from receiving duplicate alerts within a specified time. The default is 30 minutes.

In this section:

Specifying Alert Thresholds [page 1345]
Configure when alerts should be issued in the Monitor.

Suppressing Alerts for Unsubmitted Error Reports from Resources [page 1346]
Configure whether the Monitor sends out alerts when resources have unsubmitted error reports. By default, the Monitor does not send these alerts.

Related Information

Specifying Alert Thresholds [page 1345]

1.8.2.1.13.3.1 Specifying Alert Thresholds

Configure when alerts should be issued in the Monitor.

Prerequisites

> **i Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

As of version 16, to configure the alert thresholds for a database, you need the user ID and password for a user in the database that has the SYS_SAMONITOR_ADMIN_ROLE role. To configure the thresholds for an earlier database, you must have the user ID and password for a user in the database that has DBA authority.

Procedure

1. Click *Tools* > *Administration*.
2. Click *Resources*, select a resource from the list, and then click *Configure*.
3. Click *Alert Thresholds*, and then edit the threshold settings.
4. Click *Save*.
5. Click *Close*.

Related Information

Metrics [page 1307]
Alert Definitions and Thresholds [page 1342]

1.8.2.1.13.3.2 Suppressing Alerts for Unsubmitted Error Reports from Resources

Configure whether the Monitor sends out alerts when resources have unsubmitted error reports. By default, the Monitor does not send these alerts.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

Procedure

1. Log in to the Monitor.
2. Click *Tools* > *Administration*.
3. Click *Configuration*.
4. Click *Edit*.
5. Click *Options*.
6. Click *Suppress Unsubmitted Error Report Alerts From Resources*.
7. Click *Save*.
8. Click *Close*.

Results

The Monitor is configured to send alerts when resources have unsubmitted error reports.

Related Information

Troubleshooting: Reporting an Error [page 1021]

1.8.2.1.13.4 Enabling the Monitor to Send Alert Emails

Configure the Monitor to send emails to Monitor administrators and operators when alerts occur.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator and you must know information about your email protocol.

Context

The Monitor supports the SMTP and MAPI protocols for sending emails.

Procedure

1. Configure users to receive email alerts.
   a. Click *Tools* > *Administration*.
   b. Click *Users* and select your user ID.
   c. Click *Edit*, and in the *Email* field, type a valid email address.
   d. Click *Save*. Do not close the *Administration* window.
   e. On the *Users* tab, click *Email Alert Notification Settings*.
   f. Use the *Add* and *Remove* buttons to select the resources you are interested in.
   g. Click *OK*. Do not close the *Administration* window.
2. Configure the Monitor to send email alert notifications.
   a. Click the *Configuration* tab, and then click *Edit*.
   b. On the *Alert Notification* tab, click *Send alert notifications by email*.
   c. Specify the email protocol:
      - For SMTP, specify the email server and port and whether any authentication is required. When you specify a sender name and address, ensure that the sender name is a unique identifier.
      - For MAPI, specify a user ID and password.
   d. Click *Send Test Email* to test that you have properly configured email notification.
   e. Click *Save*, and then *Close*.

Related Information

Creating Monitor Users [page 1333]
Editing a Monitor User [page 1336]
Associating Monitor Users with Resources [page 1335]
Creating Monitor Users [page 1333]
Associating Monitor Users with Resources [page 1335]
Resolving and Deleting Alerts [page 1341]
Viewing Alerts [page 1340]
xp_startsmtp System Procedure

1.8.2.1.14 Backing up the Monitor

By default, the Monitor performs maintenance on the metrics once a day at midnight. Maintenance affects metrics, not alerts.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

As a Monitor administrator, you can:

- Schedule the Monitor to back up metrics.
- Control the length of time that metrics are kept.
- Perform maintenance on demand.

In this section:

Backing up Metrics [page 1349]
Back up metrics and control the length of time that metrics are kept.

Restoring a Backup Copy of the Monitor Database [page 1350]
Replace the Monitor database with a backup copy.

Related Information

Understanding How Time Is Displayed [page 1284]

1.8.2.1.14.1 Backing up Metrics

Back up metrics and control the length of time that metrics are kept.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must be a Monitor administrator.

Procedure

1. Log in to the Monitor.
2. Click *Administration*.
3. Click *Configuration*, and then click *Edit*.
4. Click *Maintenance*.
5. Specify a time (24-hour clock) when the Monitor should perform maintenance. By default, it performs maintenance at midnight. The time is local to the computer where the Monitor is running.
6. Specify a directory where the Monitor should save the backed up data. The directory must exist on the computer where the Monitor is running.
7. Customize the *Data Reduction* settings:

   Delete Metrics Older Than

   Specify a length of time. All metrics that are older than the specified length of time are deleted. The default is five days.
8. Optional. Click *Perform Maintenance Now* to run the backup and delete out-dated metrics immediately. When prompted, click *Save*.
9. Click *OK*.
10. Click *Cancel*.

Results

The daily schedule to back up and delete out-dated metrics is saved.

1.8.2.1.14.2 Restoring a Backup Copy of the Monitor Database

Replace the Monitor database with a backup copy.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must have a valid backup copy of the Monitor database.

Procedure

1. Stop the Monitor, if it is running.
2. From your backup directory, copy the `samonitor.db` database file and `samonitor.log` transaction log file.
3. Paste these files into the directory where the current Monitor database and transaction log file are located. When prompted, overwrite the existing files.

   The default locations of the Monitor database files are listed in the following table:

Operating system	Monitor directory
Microsoft Windows (Monitor installed with SQL Anywhere)	`C:\Users\Public\Documents\SQL Anywhere 17\Monitor\samonitor.db`
Microsoft Windows (Monitor installed on a separate computer)	`C:\Users\Public\Documents\SQL Anywhere 17 Monitor\samonitor.db`
Linux (Monitor installed with SQL Anywhere)	`/opt/sqlanywhere17/samonitor.db`
Linux (Monitor installed on a separate computer)	`/opt/sqlanywhere17/samonitor.db`

4. Restart the Monitor.

Results

The database is restored from a backup copy.

1.8.2.1.15 List of Database Objects Installed by the Monitor

There are several objects that are installed when you add a database as a resource to be monitored.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Object name	Object type	Description
sa_monitor_user	Database user	This read-only user is added to the database to collect metrics. Because this user is added to the database being monitored, it is not necessary to store the user credentials anywhere outside the database that is being monitored. It may be necessary to allow sa_monitor_user to bypass password verification. The sa_monitor_user has a random password known only to the Monitor. This user has the following privileges in the database: CREATE EXTERNAL REFERENCE, MONITOR, MANAGE ANY DBSPACE, BACKUP DATABASE.
sa_monitor_connection_failure	Table	This table contains metrics about failed connection attempts, and is used with sa_monitor_connection_failed_event. The metrics in this table are deleted as metrics are retrieved from the Monitor.
sa_monitor_connection_failed_event	Event	This event fires on the ConnectFailed system event (every time a connection attempt fails), and inserts a record into the sa_monitor_connection_failure table.
sa_monitor_count_unsubmitted_crash_reports	Function	This function calls the xp_srvmon_count_unsubmitted_crash_reports procedure to determine the number of unsubmitted crash reports.

Related Information

Repairing a Database Resource Monitored by the Monitor [page 1305]
Deleting Monitoring Objects from the Resource Database [page 1352]

1.8.2.1.16 Deleting Monitoring Objects from the Resource Database

Delete the monitoring objects from the resource database that were installed when you added the resource to the Monitor.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

As of version 16, to delete the monitoring objects from a database, you must have the user ID and password for a user in the database who has the SYS_SAMONITOR_ADMIN_ROLE role. To delete the monitoring objects from a pre-16.0 database, you must have the user ID and password for a user in the database that has DBA authority.

Context

Because the database objects are owned by a single owner, you can delete all of them by dropping the sa_monitor_user.

Procedure

Execute the following statement.

```
DROP USER sa_monitor_user;
```

Results

The database objects installed by the monitor are removed from the database.

1.8.2.1.17 Monitor Communications Security

You can secure communications between both the Monitor and your browser, and between the Monitor and the resources it monitors.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

In this section:

1.8.2.1.17.1 Setting up TLS Security for the Monitor

Use Transport Layer Security (TLS) to secure communication between the Monitor and your browser.

Context

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

The Monitor runs a web server that supports HTTPS connections using TLS versions 1.0 and 1.1.

Procedure

1. Obtain digital certificates from a certificate authority or create self-signed certificates with the Certificate Creation utility (createcert).
2. Alter the Monitor start line string to use the certificates.
3. Configure your browser to accept your new certificates, if required.

Results

The communication between the Monitor and your browser is secured.

Related Information

Securing the SQL Anywhere Monitor With Transport Layer Security (TLS)
Certificate Creation Utility (createcert) [page 1130]

1.8.2.1.17.2 Secure Connections Between the Monitor and Your Resources

You can use FIPS-certified encryption to encrypt communications between the Monitor and the resources it monitors.

FIPS-certified encryption requires a separate license.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

1.8.2.1.18 Troubleshooting the Monitor

There are several ways Monitor administrators can troubleshoot the Monitor.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were

previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

For example, in addition to the recommendations listed below, Monitor administrators can use the Message Log and Exception Reports features to troubleshoot the Monitor.

Problem	Recommendation
When you press F5 to refresh the browser window, you are required to log in to the Monitor.	Enable JavaScript in your browser.
You receive a network communication error when you try to log in to the Monitor.	Start the Monitor.
After upgrading to the latest version of Adobe Flash Player you continue to receive instructions to upgrade Adobe Flash Player.	Verify that the installed version Adobe Flash Player is supported by your operating system. The Monitor is backward compatible with version 10 of Adobe Flash Player.
The Monitor is unable to start monitoring a database resource.	Verify that the resource's password verification functions and login procedures allow the user sa_monitor_user to connect to the resource.
You are not receiving any alert emails.	Verify that the Monitor is properly configured to send emails and send a test email. Verify that the alert emails from the Monitor are not being blocked by a virus scanner.
The number of unscheduled requests reported by the Monitor appears to be less than the actual number of unscheduled requests.	When collecting metrics about the number of unscheduled requests, the Monitor executes query on the resource. This query could be an unscheduled request. Unscheduled queries are processed sequentially as they arrive. Therefore, if there are unscheduled requests when the Monitor attempts to execute its query, then this query must wait for the existing unscheduled requests to complete before it can execute. As a result, when the Monitor collects the number of unscheduled requests, this number does not include the unscheduled requests that existed between the time when the Monitor issued its query and the query executed.
You are not receiving alerts when the database disk space surpasses the specified threshold.	Between Monitor collection intervals, it is possible for a database to exceed the specified disk space alert threshold and the amount of space available. In such a case, the database would stop responding before the Monitor could collect the disk usage metrics and issue an alert. If your database grows quickly, set the disk space alert threshold to a higher number so that you can receive an alert before the database runs out of space.
You can't see the *Administration* window when you are logged into the Monitor.	You must be logged in to the Monitor as an administrator to have access to the *Administration* window.

Problem	Recommendation
You uninstalled the Monitor before migrating your resources and now you have lost your resources.	Uninstalling the Monitor removes the application, as well as the resources and collected metrics. When upgrading, you must install the new version of the Monitor, migrate your resources and settings, and then uninstall the old version. However, if you regularly backed up your Monitor, you can use the Migrate utility to migrate the resources from the backed up files to the new version of the Monitor.

Related Information

Alert Definitions and Thresholds [page 1342]
Monitor Users [page 1331]
Backing up the Monitor [page 1348]
Starting the Monitor [page 1271]
Enabling the Monitor to Send Alert Emails [page 1347]
Viewing the Message Log [page 1286]
Viewing Exception Reports [page 1287]
Upgrading the Monitor and Migrating Resources
xp_startsmtp System Procedure

1.8.2.1.19 Tutorial: Monitoring Resources with the Monitor

Set up monitoring for a database or MobiLink server.

Prerequisites

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You must have the user ID and password for a user in the database being monitored who has the DROP CONNECTION system privilege.

You must be logged in to the Monitor as an administrator to perform the lessons in this tutorial. Read-only and operator users can only perform a subset of the tasks that an administrator can perform. To check your Monitor user type, log in to the Monitor, click *Tools* > *User Settings*, and review the *User Type* setting.

As of version 16, to add a database, you must have the user ID and password for a user in the database who has the SYS_SAMONITOR_ADMIN_ROLE role. To add a pre-16.0 database, you must have the user ID and password for a user in the database that has DBA authority. These credentials are used to:

- Connect to the database.
- Create a new user named **sa_monitor_user**. The Monitor uses sa_monitor_user to connect to the database and monitor it.
- Install the database objects needed by the sa_monitor_user user to monitor the database.
- Discard the supplied user ID credentials from the Monitor. Because the sa_monitor_user is added to the database being monitored, it is not necessary to store the credentials of the user with the SYS_SAMONITOR_ADMIN_ROLE role or DBA authority anywhere outside the database that is being monitored.

Context

This tutorial uses the Monitor Developer Edition.

Use this tutorial to set up monitoring for a database or MobiLink server. This tutorial uses the Monitor Developer Edition.

In this section:

Related Information

Monitor Users [page 1331]

1.8.2.1.19.1 Lesson 1: Logging in to the Monitor

Log in to the Monitor to add, monitor, and collect information about resources.

Prerequisites

You must have the roles and privileges listed at the beginning of this tutorial.

Context

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Procedure

1. Start the Monitor. The Monitor must not be currently running in the background.

   To start the Monitor Developer Edition (Windows)

   Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Anywhere Monitor*.

   To start the Monitor Developer Edition (Linux)

   Run the `samonitor.sh` script from the `bin32` or `bin64` directory in the Monitor installation directory:

   ```
   ./samonitor.sh launch
   ```

   The Monitor starts collecting metrics and a browser opens the default URL where you can log in to the Monitor: http://localhost:4950.

   > **i** Note
   >
   > If you are accessing the Monitor over a network, browse to http://`computer-name`:4950, where `computer-name` is the name of the computer where the Monitor is running.

2. Log in to the Monitor as the default administrator user:
   a. In the *User Name* field, type **admin**.
   b. In the *Password* field, type **admin**.
3. By default, the Monitor opens the *Overview* dashboard that contains the following two widgets:

   Resource List

   This widget lists the resources that are being monitored. When you first open the Monitor, it is only monitoring itself via the default resource, named *SQL Anywhere Monitor*. You cannot modify this resource, nor can you stop monitoring it.

   Alert List

   This widget lists any alerts from the monitored resources.

Results

You have logged in to the Monitor as an administrator.

Next Steps

Proceed to the next lesson.

Related Information

Starting the Monitor [page 1271]

1.8.2.1.19.2 Lesson 2: Adding a Resource

Add the SQL Anywhere sample database as a resource to be monitored.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Context

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Procedure

1. Start the resource that you want to monitor. For example, to start the sample database click *Start* > *Programs* > *SQL Anywhere 17* > *SQL Anywhere* > *Network Server Sample*.
2. In the Monitor, click *Tools* > *Administration*.
3. Click *Resources*, and then click *Add*.
4. Choose one of the following options:

   Add a database resource

   1. Click *SQL Anywhere Server*, and then click *Next*.
   2. In the *Name* field, type ***demo17***, and then click *Next*.
   3. In the *Host* field, type **`localhost`**, and in the *Server* field, type ***demo17***.
   4. Click *Create*.
   5. When you are prompted for the required authorization, in the *User ID* field, type **`DBA`**, and in the *Password* field, type **`sql`**. Click *OK*.
      The resource is added and monitoring the resource starts automatically.
5. Click *Close* twice.
6. Click *Dashboards* > *Overview*.

   The resource appears in the *Resource List*.
7. Click the resource to open the dashboard and view the collected metrics.

Results

The resource is added and monitored.

Next Steps

Proceed to the next lesson.

Related Information

Adding a Database Resource [page 1289]
Adding a MobiLink Server Resource [page 1291]
List of Database Objects Installed by the Monitor [page 1351]
Host Connection Parameter [page 86]

1.8.2.1.19.3 Lesson 3: Testing an Alert

Trigger an alert so that you can practice handling alerts.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Context

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Procedure

1. Trigger an alert by shutting down the database server.

   For example, shut down the database server on Windows by double-clicking the network server icon in the system tray, and then clicking *Shut Down* in the database server messages window.
2. In the Monitor, click *Dashboards* > *Overview*.

   In the *Resource List*, a red circle with a white X through it appears next to the resource to indicate that the resource is unavailable.

   In the *Alert List* widget, an *Availability Alert* for a database appears with the status *Active*.

   It can take a few seconds for these changes in state and status to occur. By default, the Monitor collects information from the resource every 30 seconds.

3. In the *Alert List* widget, click the alert to read its description.
4. Click *OK* to close the alert.
5. Restart the resource. For example, restart the sample database by clicking *Start* > *Programs* > *SQL Anywhere 17* > *SQL Anywhere* > *Network Server Sample*.

   In the *Resource List* widget, the *Status* for the resource changes to a yellow triangle. This icon indicates that the resource is being monitored and it has an alert. In the *Alert List*, the *Status* of the alert changes to *Inactive*. An inactive alert indicates that the issue that triggered the alert is no longer present, but the alert has not been resolved or deleted.
6. Resolve the alert by selecting the alert and clicking *Mark Resolved*.

   In the *Resource List*, there are no icons beside the resource, indicating that the resource is being monitored and it has no alerts.

Next Steps

Proceed to the next lesson.

Related Information

Metrics [page 1307]
Monitor Users [page 1331]

1.8.2.1.19.4 Lesson 4: Adding a New Dashboard and Widgets

Add a dashboard and widgets to the Monitor to display information about resources.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Context

Dashboards are user-specific. Any user can add, edit, or delete their own dashboards and the widgets that exist in their dashboards.

> **i Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Procedure

1. Click *Dashboards* > *Add New*.
   a. Click *A Dashboard To Monitor The Following Resource*, and choose the resource.
   b. In the *Dashboard Name* field, type **user_joe**.
   c. In the *Number Of Columns* field, type **2**.
   d. Click *OK*.

   A new dashboard appears, with the following widgets:

   For databases

   Alert List, *Key Performance Metrics*, *Server Info*, and *Connections*.

   For MobiLink

   Alert List, *Key Performance Metrics*, *Server Info*, and *Connections*.
2. Add a metrics widget to display graphs instead of the default spark lines.
   a. In the upper right corner of the `user_joe` dashboard, click *Customize* > *Add Widget*.
   b. Click *Metrics*, and then click *Next*.
   c. In the *What Do You Want To Name This Widget?* field, type `Metrics display`.
   d. In the *Which Resource Are You Interested In?* field, click *demo17* for a database server, *MobiLinkServerSample* for MobiLink server
   e. In the *What Kind Of Display Do You Want?* field, click *Graph*.
   f. For *What Metrics Do You Want To See?*, click one of the following:

      For databases

      CPU Usage, *Queries Processed*, and *Memory Metrics* > *Cache Size*.

      For MobiLink

      Synchronization Metrics
   g. Click *Create*.

   A widget called *Metrics display* appears in the dashboard.

   To maximize the size of a widget, in the widget pane, click the dropdown menu arrow, and click *Maximize*.

   To view details on the graph, position your cursor above specific points on the graph.

Results

You have created a dashboard and a metrics widget for your resource.

Next Steps

Proceed to the next lesson.

1.8.2.1.19.5 Lesson 5: Closing Database Connections

Close a connection to a resource database.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Context

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Procedure

1. Open the *demo17* dashboard.
2. In the *Connections* widget, click the *x* beside the name of the connection you want to close.
3. When you are prompted for the required authorization, in the *User ID* field, type **`DBA`**, and in the *Password* field, type **`sql`**.

   The Monitor uses the user ID and password to connect to the resource database and drop the connection. This user ID must have DROP CONNECTION system privilege. The user ID and password are not kept by the Monitor.

4. Click *OK*.

   The connection is removed from the *Connections* widget.

Results

A connection is closed.

Next Steps

Proceed to the next lesson.

1.8.2.1.19.6 Lesson 6: Stopping Monitoring of Resources

Remove the resource you were monitoring, delete the collected metrics for it, and stop data collection on it.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Context

> **i Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

You cannot delete the *SQL Anywhere Monitor* resource.

> **i Note**
>
> When you remove a database resource, the Monitor does *not* delete the monitoring objects installed in the database. To delete these objects, you must connect to the database as a user with MANAGE ANY USER system privilege, and execute the following statement:
>
> ```
> DROP USER sa_monitor_user;
> ```

Procedure

1. Remove the resources that you added as part of this tutorial.
   a. Click *Tools* > *Administration*.
   b. Click *Resources*.
   c. Select either the *demo17*, or the *MobiLinkServerSample* and click *Stop*.
   d. Click *Remove*.
   e. Click *Yes* to confirm that you want to remove the resource.
   f. Click *Close*.
2. Click *Logout* to log out of the Monitor and close the browser window where you are viewing the Monitor.
3. Shut down the Monitor and stop monitoring.
   - On Windows, in the system tray, right-click the Monitor icon and click *Exit SQL Anywhere Monitor*.
   - On Linux, run the `samonitor.sh` script from the `bin32` or `bin64` directory in the Monitor installation directory:

     ```
     ./samonitor.sh stop
     ```

   The Monitor stops collecting metrics for all resources.
4. Shut down the resource that was being monitored. For example, to shut down the database double-click the network server icon in the system tray for the database server, click *Shut Down* in the database server messages window, and then click *Yes* to confirm the action.

Results

The Monitor is shut down and stops monitoring the resource, which is also shut down.

Next Steps

Proceed to the next lesson.

Related Information

Monitor Users [page 1331]

1.8.2.1.19.7 Lesson 7: (Optional) Monitoring a Database Mirroring System

Use the Monitor to monitor a database mirroring system.

Prerequisites

This tutorial assumes that you have a running read-only scale-out system as described in the tutorial for creating a database mirroring system.

You must have the roles and privileges listed at the beginning of this tutorial.

Context

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Procedure

1. Add a database mirroring system to the Monitor's *Resource List*.
   a. In the *Tools* sidebar, click *Administration*.
   b. Click *Resources*, and then click *Add*.
   c. Click *SQL Anywhere Server*, and then click *Next*.
   d. In the *Name* field, type **`mirror_demo`**, and then click *Next*.
   e. In the *Host* field, specify the host names and port numbers of the computers running the primary and mirror servers in a comma separated list. That is, type **`localhost:6871, localhost:6872`**.
   f. In the *Server* field, specify the alternate server name for the primary server. That is, type **`mirror_demo_primary`**.
   g. Click *Create*.
   h. When you are prompted for the required authorization, type **`DBA`** in the *User ID* field and **`sql`** in the *Password* field. Click *OK*.

      The resource is added and monitoring of the resource starts automatically.
   i. Click *Close* twice.
2. Add a widget to monitor `scaleoutdemo.db`:
   a. In the *Dashboard* sidebar, click *mirror_demo*.

b. In the top right corner of the mirrorsystemdemo dashboard, click *Customize* > *Add Widget*.
c. Select *SQL Anywhere Scale-Out Topology* and click *Next*.
d. In the *What Do You Want To Name This Widget?* field, type **`mirror_demowidget`**.
e. In the *Which Resource Are You Interested In?* field, select **`mirror_demo`**.
f. Click *Create*.

The new widget appears in the dashboard.

Results

A database mirroring system resource is added and monitored.

Next Steps

Proceed to the next lesson.

Related Information

SQL Anywhere Monitor [page 1265]
Dashboards and Widgets [page 1277]
SQL Anywhere Scale-out Topology Widget [page 1281]
Tutorial: Creating a Database Mirroring System [page 1800]
Adding a Database Resource [page 1289]
Adding a MobiLink Server Resource [page 1291]
Managing Widgets [page 1280]

1.8.2.1.19.8 Lesson 8: (Optional) Monitoring Your Read-only Scale-out System from the SQL Anywhere Monitor

Start a database that is the root node in a read-only scale-out system and then add this database as a resource in the Monitor.

Prerequisites

You must have completed the previous lessons in this tutorial to set up a running read-only scale-out system.

You must have the roles and privileges listed at the beginning of this tutorial.

Context

> **i Note**
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Procedure

1. Start the Monitor. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Anywhere Monitor*.
2. Log in to the Monitor as the default administrator user:
   a. In the *User Name* field, type **admin**.
   b. In the *Password* field, type **admin**.
3. Add `scaleoutdemo.db` as a resource to the Monitor:
   a. In the *Tools* sidebar, click *Administration*.
   b. Click *Resources*, and then click *Add*.
   c. Click *SQL Anywhere Server*, and then click *Next*.
   d. In the *Name* field, type **scaleoutdemo**, and then click *Next*.
   e. In the *Host* field, type **localhost**.
   f. In the *Server* field, type **scaleout_root_demo**.
   g. Click *Create*.
   h. When you are prompted for the required authorization, type **DBA** in the *User ID* field and **sql** in the *Password* field. Click *OK*.

      The resource is added and monitoring of the resource starts automatically.
   i. Click *Close* twice.
4. Add a widget to monitor `scaleoutdemo.db`:
   a. In the *Dashboard* sidebar click *scaleoutdemo*.
   b. In the top right corner of the scaleoutdemo dashboard, click *Customize* > *Add Widget*.
   c. Select *SQL Anywhere Scale-Out Topology* and click *Next*.
   d. In the *What do you want to name this widget* field, type **scaleoutwidget**.
   e. In the *Which resource are you interested in* field, click *scaleoutdemo*.
   f. Click *Create*.

   The new widget appears in the dashboard.

Results

You have added your read-only scale-out system as a resource to the Monitor, created a widget to monitor your read-only scale-out system, and collect metrics.

Related Information

SQL Anywhere Monitor [page 1265]
Dashboards and Widgets [page 1277]
SQL Anywhere Scale-out Topology Widget [page 1281]
Tutorial: Creating a Read-only Scale-out System [page 1853]
Adding a Database Resource [page 1289]
Adding a MobiLink Server Resource [page 1291]
Managing Widgets [page 1280]

1.8.2.2 Solution Manager

SAP Solution Manager provides tools that you can use to monitor SQL Anywhere as part of your overall SAP landscape.

Configuration file

The `ncs.conf` file provides all of the necessary connection and configuration information required to deliver monitoring data to the SAP Solution Manager. To enable monitoring, use the -ncs database server option. Use the -ncsd database server option to specify the location of the `ncs.conf` configuration file.

Related Information

-ncs Database Server Option [page 469]
-ncsd Database Server Option [page 470]
Maintenance of Product in the System Landscape
SAP Solution Manager Setup

1.8.2.3 The SQL Anywhere SNMP Extension Agent

You can use the SQL Anywhere SNMP Extension Agent with SNMP management applications to manage your databases.

One agent can be used to monitor several different databases running on different database servers running on different computers.

Using the SQL Anywhere SNMP Extension Agent, you can:

- Retrieve the value of all server and database statistics.
- Retrieve the value of all server and database properties.
- Retrieve the value of all PUBLIC database options.
- Set the value for any PUBLIC database option.
- Execute stored procedures.
- Generate traps based on property or statistic values.

Supplied Files

The following files for the SQL Anywhere SNMP Extension Agent are included in your SQL Anywhere installation:

dbsnmp17.dll

The SQL Anywhere SNMP Extension Agent. This file is located in *%SQLANY17%*`\bin32`.

SQLAnywhere.mib

The SQL Anywhere MIB contains all the OIDs for database server and database properties, statistics, and options that can be accessed using the SQL Anywhere SNMP Extension Agent.

RDBMS-MIB.mib

This is a generic MIB for relational database management systems and contains OIDs that can be accessed using the SQL Anywhere SNMP Extension Agent.

SNMPv2-SMI.mib

This MIB is referenced by the SQL Anywhere and RDBMS MIBs.

SNMPv2-TC.mib

This MIB is referenced by the SQL Anywhere and RDBMS MIBs.

SYBASE-MIB.mib

The Sybase MIB. This MIB is referenced by the SQL Anywhere MIB.

sasnmp.ini

This file lists the databases that the SQL Anywhere SNMP Extension Agent monitors. By default, this file is located in *%SQLANY17%*`\bin32`.

To use the SQL Anywhere SNMP Extension Agent, you must have SNMP installed on your computer and you must create an `sasnmp.ini` file that contains information about the databases that are monitored by the SQL Anywhere SNMP Extension Agent.

In this section:

1.8.2.3.1 SNMP

Simple Network Management Protocol (SNMP) is a standard protocol used for network management.

SNMP allows **managers** and **agents** to communicate: managers send requests to agents, and agents respond to queries from managers. Additionally, agents can notify managers when specific events occur using notifications called **traps**.

SNMP agents handle requests to get and set the values of variables for managed objects. Each variable has a single value, and values are generally strings or integers, although they may also be other types.

Variables are kept in a global hierarchy, and each variable has a unique number under its parent. The full name of a variable (including all its parents) is called the **Object Identifier** (OID). All OIDs that are owned by SAP begin with 1.3.6.1.4.1.897.

The list of OIDs that an agent supports, including their names, types, and other information are stored in a file called a **Management Information Base** (MIB).

A MIB is a database that stores network management information about managed objects. The MIB is separate from the SQL Anywhere database you are monitoring using the SQL Anywhere SNMP Extension Agent. The values of MIB objects can be changed or retrieved using SNMP. MIB objects are organized in a hierarchy with the most general information about the network located at the top level of the hierarchy. The SQL Anywhere SNMP Extension Agent supports the SQL Anywhere MIB and the RDBMS MIB.

In this section:

1.8.2.3.1.1 The SQL Anywhere MIB

The SQL Anywhere MIB was created for the SQL Anywhere SNMP Extension Agent, and includes all database server statistics and properties, and all database statistics, properties, and options.

The statistics and properties are all read-only (with a few exceptions), and the database options are all read-write.

By default, the SQL Anywhere MIB is located in *%SQLANY17%*`\snmp\iAnywhere.mib`.

The following hierarchy describes the SQL Anywhere MIB:

OID	Name	Description
1.3.6.1.4.897.2.1.1.n.db	saServer.saSrvStat	Returns the value of server statistic n on database db.
1.3.6.1.4.897.2.1.2.n.db	saServer.saSrvProp	Returns the value of server property n on database db.
1.3.6.1.4.897.2.2.1.n.db	saDb.saDbStat	Returns the value of database statistic n on database db.
1.3.6.1.4.897.2.2.2.n.db	saDb.saDbProp	Returns the value of database property n on database db.
1.3.6.1.4.897.2.2.3.n.db	saDb.saDbOpt	Returns the value of database option n on database db.
1.3.6.1.4.897.2.3.1	saAgent.saVersion	Returns the version of the SQL Anywhere SNMP Extension Agent.
1.3.6.1.4.897.2.3.2.db	saAgent.saDbConnStr	Returns the connection string for database db.
1.3.6.1.4.897.2.3.3.db	saAgent.saConnected	Returns whether the SQL Anywhere SNMP Extension Agent is connected to database db. Setting this value to 0 causes the SQL Anywhere SNMP Extension Agent to disconnect from the database, while setting this value to 1 causes the SQL Anywhere SNMP Extension Agent to attempt to connect to the database.

OID	Name	Description
1.3.6.1.4.897.2.3.4.db	saAgent.saStarted	Returns whether database db is running. Setting this value to 0 causes the SQL Anywhere SNMP Extension Agent to shut down the database[1], while setting this value to 1 attempts to start the database[2].
1.3.6.1.4.897.2.3.5.db	saAgent.saProc	Setting this value to a string proc_name causes the SQL Anywhere SNMP Extension Agent to executed the procedure proc_name in the database. Arguments can be supplied (for example, proc_name('string', 4)); if no arguments are supplied, parentheses () are appended to the name. Getting the value returns empty quotations ("").
1.3.6.1.4.897.2.3.6	saAgent.saRestart	Setting the value of this variable to 1 causes the agent to restart itself (it disconnects from all databases and reloads the .ini file). Getting the value returns 0.
1.3.6.1.4.897.2.3.7	saAgent.salnifile	Returns the full path of the sasnmp.ini file the SQL Anywhere SNMP Extension Agent is using.
1.3.6.1.4.897.2.4	saMetaData	Several virtual tables; each row represents a variable supported by the SQL Anywhere MIB.

[1] When stopping a database by setting this variable, the stop is unconditional, meaning that the database will be stopped even if it has active connections.

[2] To be able to start a database by setting this variable, the DBF parameter must be specified in the connection string (including the DBN, and DBKEY if it is required), and either the *UtilDbPwd* field must be set in the sasnmp.ini file, or the start database privilege on the server (specified with the -gd server option) must be set to all.

saMetaData Tables

The SQL Anywhere MIB includes metadata tables that provide a way to query the SQL Anywhere SNMP Extension Agent to find out which variables are supported.

saSrvMetaData.saSrvStatMetaDataTable

Lists the database server statistics (variables under sa.saServer.saSrvStat).

saSrvMetaData.saSrvPropMetaDataTable

Lists the database server properties (variables under sa.saServer.saSrv.Prop).

saDbMetaData.saDbStatMetaDataTable

Lists the database statistics (variables under sa.saDb.saDbStat).

saDbMetaData.saDbpropMetaDataTable

Lists the database properties (variables under sa.saDb.saDbProp).

saDbMetaData.saDbOptMetaDataTable

Lists the database options (variables under sa.saDb.saDbOpt).

Related Information

SQL Anywhere MIB Reference [page 1386]
Value Setting Using the SQL Anywhere SNMP Extension Agent [page 1381]
saMetaData Tables [page 1387]

1.8.2.3.1.2 The RDBMS MIB

The RDBMS MIB is a generic and vendor-independent MIB (RFC 1697) for relational database management system products.

The RDBMS MIB uses **virtual tables** to return information about the servers and databases. The base OID is 1.3.6.1.2.1.39, and there are 9 virtual tables in this MIB. The SQL Anywhere SNMP Extension Agent supports eight of these virtual tables.

The SQL Anywhere SNMP Extension Agent provides read-only access to all the supported variables in the RDBMS MIB. None of the variables in the RDBMS MIB are writable through the SQL Anywhere SNMP Extension Agent.

A virtual table contains a fixed number of attributes and any number for rows. Elements in the table are retrieved using `GET` requests by appending the column number and row number to the OID of the table. A 1 must be appended to the table OID, so the OID looks as follows:

```
table.1.column.rownum
```

By default, the RDBMS MIB is located in *%SQLANY17%*`\snmp\RDBMS-MIB.mib`.

1.8.2.3.2 SNMP Installation

To use the SQL Anywhere SNMP Extension Agent, you must install SNMP on your computer.

By default, SNMP is not installed on Windows.

For information about installing SNMP, see your operating system documentation.

Once you install SNMP on your computer, the following services should be running on your computer: SNMP Service and SNMP Trap Service.

If you installed SNMP before you installed SQL Anywhere, you need to stop and restart the SNMP service so it can detect the SQL Anywhere SNMP Extension Agent. If you installed SQL Anywhere and then installed SNMP, the SNMP service detects the SQL Anywhere SNMP Extension Agent automatically.

In this section:

1.8.2.3.2.1 Restarting the SNMP Service

Restart the SNMP service so that it can detect the SQL Anywhere SNMP Extension Agent.

Prerequisites

You must be an administrator on your computer.

Context

You must restart the service if you installed SNMP before you installed SQL Anywhere.

Procedure

1. Run the following command to stop the SNMP service:

   ```
   net stop snmp
   ```

2. Run the following command to start the SNMP service:

   ```
   net start snmp
   ```

Results

The SNMP service is restarted.

1.8.2.3.2.2 Restarting the SNMP Extension Agent

Restart the SNMP Extension Agent.

Prerequisites

You must be an administrator on your computer.

Context

You must restart the agent when the contents of the `sasnmp.ini` file change.

Procedure

Using your SNMP management tool, change the value of the saAgent.saRestart property, `1.3.6.1.4.1.897.2.3.6`, to 1.

Results

The changes to the SNMP Extension Agent take effect.

Next Steps

You can encrypt the contents of the `sasnmp.ini` file using the File Hiding utility (dbfhide). If you do so, make sure to keep an unencrypted copy of the file so that you will be able to update the encrypted file.

Related Information

Hiding the Contents of an .ini File [page 591]

1.8.2.3.3 SQL Anywhere SNMP Extension Agent Configuration

The SQL Anywhere SNMP Extension Agent can monitor one or more databases.

The databases to be monitored are stored in the `sasnmp.ini` file with the following format:

```
[SAAgent]
TrapPollTime=time-in-seconds
[DBn]
ConnStr=connection-string
UtilDbPwd=utility-database-password
CacheTime=time-in-seconds
DBSpaceCacheTime=time-in-seconds
Trapt=trap-information
Disabled=1 or 0
```

By default, your SQL Anywhere installation places the `sasnmp.ini` file in the *%SQLANY17%*`\bin32` directory.

You can encrypt the contents of the `sasnmp.ini` file using the File Hiding utility (dbfhide).

The SAAgent Section

The SAAgent section of the `sasnmp.ini` file contains information about the SQL Anywhere SNMP Extension Agent. If the TrapPollTime property is not required, you can omit the entire section.

TrapPollTime

This value specifies the poll frequency for dynamic traps if they are specified. The SQL Anywhere SNMP Extension Agent polls the values every 5 seconds by default. Setting this value to 0 disables dynamic traps. This property is optional.

The DB`n` section

Each DB`n` section of the `sasnmp.ini` file describes a database, how to connect to it, and any dynamic traps that exist for the database. The property names in the file are case sensitive.

The value for `n` is a number that identifies the database. The numbers must start with 1, and numbers cannot be skipped. For example, if the `sasnmp.ini` file contained entries for [DB1], [DB2], and [DB4], the [DB4] entry would be ignored because the file is missing the entry for [DB3].

ConnStr

The connection string used to connect to the database. You must supply enough information to be able to connect to the database. This property is required.

- To use an ODBC data source to connect to the database, it must be a *system* data source, not a *user* data source.
- To use an integrated login, you must map to the SYSTEM account because the SNMP Agent runs as a service. However, anything that runs as a service can then connect to the database without a

password. Alternatively, you can change the account that the service runs under and then create an integrated login for that account.

- The string `ASTART=NO;IDLE=0;CON=SNMP;ASTOP=NO` is prepended to the connection string. This string does the following:
  - prevents the SQL Anywhere SNMP Extension Agent from trying to start a database server automatically
  - disables idle timeout since it is likely that the SQL Anywhere SNMP Extension Agent will sit idle for some time
  - names the connection so it can be identified
  - prevents the database from being shut down when the SQL Anywhere SNMP Extension Agent disconnects

  If you specify any of these values in the connection string in the `sasnmp.ini` file, the values in the `sasnmp.ini` file will override the default settings.

UtilDbPwd

When setting `sa.agent.saStarted` to start a database, the SQL Anywhere SNMP Extension Agent attempts to connect to the database with the DBF parameter, which tells the database server where to find the database file. However, if the privilege required to start the database is DBA (the default for the network server, which can also be set using the -gd dba option for both the personal and network servers), then the server will not allow the connection.

To start a database on such a server, the SNMP Extension Agent must connect as a user with the SERVER OPERATOR system privilege to a database already running on the same server. This can be done by connecting to the utility database. If you specify the utility database password (specified by the -su server option) in the `sasnmp.ini` file, then to start a database, the SQL Anywhere SNMP Extension Agent connects to the utility database on the same server, executes the START DATABASE statement, and then disconnects. This property is optional.

CacheTime

When data is retrieved from the database, it can be cached inside the SQL Anywhere SNMP Extension Agent, so that subsequent retrievals of the same type of data (for example, server properties or database statistics) do not require communication with the database. While caching the data means that you can obtain the data more quickly on subsequent retrievals, the data may be out of date. The CacheTime property can be used to change the cache time, or disable the cache by setting the value to 0. By default, the cache time is 0 seconds. When the CacheTime parameter is set to 0, the data retrieved is always up-to-date because data is retrieved from the database for every request. This property is optional.

DBSpaceCacheTime

The rdbmsDbLimitedResourceTable in the RDBMS MIB contains information about dbspaces. When this information is retrieved from the database, it can also be cached inside the SQL Anywhere SNMP Extension Agent. The default cache time for dbspace information is 600 seconds (10 minutes). This property can be used to change the cache time (or disable the cache by setting the value to 0). This property is optional.

Trap t

Creates a dynamic trap. The value `t` must be a positive integer starting at 1. Skipping numbers is not allowed. This property is optional.

Disabled

If set to 1, this database entry is skipped by the SQL Anywhere SNMP Extension Agent. This is useful for temporarily removing one database from the list of databases managed by the SQL Anywhere SNMP Extension Agent, without renumbering the rest. This property is optional.

Once you edit this file, you must restart the SNMP service or reset the SQL Anywhere SNMP Extension Agent so that the new settings are used by the Agent.

Sample sasnmp.ini File

The following is a sample `sasnmp.ini` file for the SQL Anywhere SNMP Extension Agent.

```
[SAAgent]
[DB1]
ConnStr=UID=DBA;PWD=passwd;DBN=sales;DBF=sales.db
Trap1=1.1.5 > 50000
UtilDbPwd=test
[DB2]
ConnStr=UID=DBA;PWD=passwd;DBN=field;DBF=field.db
UtilDbPwd=test
Disabled=1
[DB3]
ConnStr=UID=DBA;PWD=passwd;Host=host2;DBN=hq;DBF=hq.db
UtilDbPwd=test
```

Because there are no parameters specified in the SAAgent section, the SQL Anywhere SNMP Extension Agent will poll values every 5 seconds.

The SQL Anywhere SNMP Extension Agent is monitoring 3 different databases running on two different servers. Database 3 is running on a different computer, so the Host connection parameter is required. A trap is specified for DB1, which fires when the number of bytes sent by the database server is greater than 50000.

Related Information

Dynamic Traps [page 1384]
rdbmsDbLimitedResourceTable [page 1416]

1.8.2.3.4 Values Returned by the SQL Anywhere SNMP Extension Agent

Use the SQL Anywhere SNMP Extension Agent to retrieve the values of database server properties, database server statistics, database options, database properties, and database statistics.

The way you retrieve these values depends on your SNMP management software.

Example

The table below provides a description and sample value that could be returned for the following OIDs.

OID	Explanation	Sample value
1.3.6.1.4.1.897.2.1.1.1.1	Server statistic ActiveReq on database 1	1
1.3.6.1.4.1.897.2.2.1.4.1	Database statistic CacheRead on database 1	11397
1.3.6.1.4.1.897.2.3.1	Agent version	17.0.11(2459)
1.3.6.1.4.1.897.2.3.2.1	Connection string for database 1	UID=DBA;PWD=passwd; Host=host1; DBN=sales; DBF=sales.db

Related Information

SQL Anywhere MIB Database Server Properties [page 1393]
SQL Anywhere MIB Database Server Statistics [page 1390]
SQL Anywhere MIB Database Options [page 1407]
SQL Anywhere MIB Database Properties [page 1403]
SQL Anywhere MIB Database Statistics [page 1399]

1.8.2.3.5 Value Setting Using the SQL Anywhere SNMP Extension Agent

You can set any database option, some server properties, and one database property using the SQL Anywhere SNMP agent.

The SQL Anywhere SNMP Extension Agent responds to SNMP get, get-next, and set queries.

When setting database options, the SQL Anywhere SNMP agent executes the statement:

```
SET OPTION PUBLIC.option-name = 'value'
```

When setting database and server properties, the sa_server_option system procedure is used.

The way you set these values depends on your SNMP management software.

There are many options and properties that can be set using the SQL Anywhere MIB files, provided with the SQL Anywhere SNMP Extension Agent.

Related Information

Database Options [page 651]

SQL Anywhere MIB Reference [page 1386]
SET OPTION Statement
sa_server_option System Procedure

1.8.2.3.6 Stored Procedures and the SQL Anywhere SNMP Extension Agent

The SQL Anywhere MIB includes an OID that allows you to execute a stored procedure using the SQL Anywhere SNMP Extension Agent.

To execute the stored procedure, the user that the SQL Anywhere SNMP Extension Agent uses to connect must have one of the following:

- EXECUTE privilege on the procedure
- be the owner of the procedure
- have the EXECUTE ANY PROCEDURE system privilege

Any result sets or return values generated by the procedure are ignored.

To execute a stored procedure using the SQL Anywhere SNMP Extension Agent, set the value of `saAgent.saProc` (OID 1.3.6.1.4.1.897.2.3.5.`db`, where `db` is the database number in the `sasnmp.ini` file) to a string that is the name of a stored procedure. Optionally, you can supply arguments to the procedure; if no arguments are supplied, parentheses are appended to the procedure name.

For example, setting the value of `saAgent.saProc` to the string "`pchin.updatesales( 'param1', 2)`" calls the updatesales stored procedure owned by user pchin.

The way you set the value of this OID to the procedure name depends on your SNMP management software.

Related Information

The SQL Anywhere MIB [page 1373]

1.8.2.3.7 Traps

A **trap** is an OID that is sent by an SNMP agent when a particular event occurs.

Traps are initiated by the SNMP agent and can be detected by SNMP management software, which can then either deal with the event directly or query the SNMP agent for more information.

To receive traps, you must configure the SNMP service. The SNMP service will receive the trap information and then forward it on somewhere; however, by default, this is nowhere, so any trap listeners you have running will not detect anything.

SQL Anywhere SNMP Extension Agent Traps

The SQL Anywhere SNMP Extension Agent sends a trap whenever a connection is dropped by the database server. The OID of this trap is `1.3.6.1.2.1.39.2.1`.

If you are using database mirroring, and the SQL Anywhere SNMP Extension Agent connection to the database server drops, every 30 seconds the SQL Anywhere SNMP Extension Agent attempts to reconnect to the database server. When the agent reconnects, if it finds that it is connected to a different database server (as determined by the ServerName property), then it sends a trap with the OID 1.3.6.1.4.1.897.2.6.3, and the database ID from the `sasnmp.ini` file. In this case, the SQL Anywhere SNMP Extension Agent was connected to the primary database server, which went down, and now the mirror server is acting as the primary server.

The only other traps sent by the SQL Anywhere SNMP Extension Agent are dynamic traps.

In this section:

Related Information

Database Mirroring [page 1757]

1.8.2.3.7.1 Configuring the SNMP Service to Send Traps to Your Computer

Configure the SNMP service to send notifications about events to your computer by using traps.

Prerequisites

You must be an administrator on your computer.

Procedure

1. Right-click *My Computer* and click *Manage*.
2. In the left pane, double-click *Services And Applications*.

3. In the left pane, double-click *Services*.
4. Locate SNMP Service in the list of services in the right pane, right-click it and click *Properties*.
5. Click the *Traps* tab.
6. Click *Add*.
7. In the *SNMP Service Configuration* window, type **`localhost`** in the text box and then click *Add*.
8. Click *OK*.

Results

A trap is added to the SNMP service to send notifications to your computer.

1.8.2.3.7.2 Dynamic Traps

A **dynamic trap** is a trap that is sent by the SQL Anywhere SNMP Extension Agent when a simple expression involving the value of a particular property, statistic, or option is true.

Dynamic traps are created in the `sasnmp.ini` file. The format of the trap information in the `sasnmp.ini` file entry is as follows:

```
Traptrapnum=[1.3.6.1.4.1.897.2.]oid[.dbnum] op value
```

trapnum

is the dynamic trap number. It must start at 1 and be sequential.

oid

is the OID of the property, statistic, or option. OIDs in either the SQL Anywhere MIB or the RDBMS MIB are supported. If the OID given is an invalid SQL Anywhere or RDBMS OID, the SQL Anywhere MIB prefix (1.3.6.1.4.1.897.2.) is prepended.

> **i** Note
>
> You can only use OIDs corresponding to database server or database properties, statistics, or options in dynamic traps.

dbnum

is the database number. This field is optional, but if specified, it must match the database number of the [DB*n*] section of the `sasnmp.ini` file.

must have one of the following values:

- = or == (equality)
- *!=*, <>, or >< (inequality)
- <= or =< (less than or equal)
- >= or => (greater than or equal)
- < (less than)

- > (greater than)

op

> **Note**
>
> Only equality or inequality is supported for string values.

value

is the value to use in the expression. String values may be enclosed in single or double quotes; these quotes are not included in the value. If you want the beginning or closing quotation marks to be included in the string, you must double them. Single quotes occurring within the string should not be doubled.

When setting dynamic traps, use k, m, g, or t to specify units of kilobytes, megabytes, gigabytes, or terabytes. For example, you can set a dynamic trap to fire if the current cache size exceeds 200 MB by using:

```
Trap1=1.3.6.1.4.1.897.2.1.1.11.1 > 200M
```

You can specify as many Trap fields as you want in the `sasnmp.ini` file. The OID used for the trap is `1.3.6.1.4.1.897.2.4.1`, and the data sent with the trap includes the following:

- the trap number (starts at 1 for the first dynamic trap sent by the SQL Anywhere SNMP agent)
- the database index
- the database name trap index (from the `sasnmp.ini` file)
- the variable name
- the variable value (this is the current value of the variable, not necessarily the threshold value)

Dynamic Trap Behavior

Once a dynamic trap is triggered, the trap is not sent again until the condition that caused it to be triggered changes to FALSE and then back to TRUE again.

For example, if you have a dynamic trap set using 1.1.11.1 >= 51200K, then the trap is triggered when the server's cache size reaches 50 MB (= 51200 KB) and the dynamic trap is disabled, so no more traps are sent. The only way the trap is re-enabled is if the cache size later drops below 50 MB. You would then be notified if the cache size grew to 50 MB again.

Trap Examples

Trap information	Description
Trap1=1.1.5 > 10000	Trap sent when the number of bytes sent from the server is greater than 10000.
Trap2=1.3.6.1.2.1.39.1.4.1.4.14.1 >= 10485760	Trap sent if the size of the transaction log file is larger than 10 MB.

Related Information

SQL Anywhere MIB Reference [page 1386]
RDBMS MIB Reference [page 1413]

1.8.2.3.8 SQL Anywhere MIB Reference

The list of object identifiers (OIDs) that an SNMP agent supports, including their names, types, and other information are stored in a file called a Management Information Base (MIB).

There are many statistics, properties, and options that can be retrieved and set using the SQL Anywhere SNMP Extension Agent.

In this section:

Related Information

SNMP [page 1372]

1.8.2.3.8.1 Agent

The Agent table lists information about the SQL Anywhere SNMP Extension Agent.

Writable properties are marked with an asterisk (*). The value `n` is the database number in the `sasnmp.ini` file.

OID	Type	Name	Value returned
1.3.6.1.4.1.897.2.3.1	String	saVersion	Agent version
1.3.6.1.4.1.897.2.3.2.n	String	saDBConnStr	Connection string
1.3.6.1.4.1.897.2.3.3.n	Integer32	saConnected*	1 if the agent is connected, 0 otherwise
1.3.6.1.4.1.897.2.3.4.n	Integer32	saStarted*	1 if the database is started, 0 otherwise
1.3.6.1.4.1.897.2.3.5.n	String	saProc*	" "
1.3.6.1.4.1.897.2.3.6	String	saRestart*	0

1.8.2.3.8.2 saMetaData Tables

There are several metadata tables that are included in the SQL Anywhere MIB.

- saSrvMetaData.saSrvStatMetaDataTable
- saSrvMetaData.saSrvPropMetaDataTable
- saDbMetaData.saDbStatMetaDataTable
- saDbMetaData.saDbPropMetaDataTable
- saDbMetaData.saDbOptMetaDataTable

In this section:

1.8.2.3.8.2.1 saSrvMetaData.saSrvStatMetaDataTable

Contains metadata about the database server statistics.

The value `db` is the database number in the `sasnmp.ini` file.

OID	Type	Name	Value returned
1.3.6.1.4.1.897.2.4.1.1.1.1.db	Integer32	saSrvStatIndex	db
1.3.6.1.4.1.897.2.4.1.1.1.2.db	Integer32	saSrvStatObjType	1[1]
1.3.6.1.4.1.897.2.4.1.1.1.3.db	Integer32	saSrvStatType	1[2]
1.3.6.1.4.1.897.2.4.1.1.1.4.db	OID	saSrvStatOID	OID of SQL Anywhere MIB entry[3]
1.3.6.1.4.1.897.2.4.1.1.1.5.db	String	saSrvStatName	Statistic name

[1] Values: 1=Server, 2=Database

[2] Values: 1=Statistic, 2=Property, 3=Option

[3] The OID returned does not include the database number. You must append the database number to the OID before it can be used in a query.

1.8.2.3.8.2.2 saSrvMetaData.saSrvPropMetaDataTable

Contains metadata about the database server properties.

The value db is the database number in the `sasnmp.ini` file.

OID	Type	Name	Value returned
1.3.6.1.4.1.897.2.4.1.2.1.1.db	Integer32	saSrvPropIndex	db
1.3.6.1.4.1.897.2.4.1.2.1.2.db	Integer32	saSrvPropObjType	1[1]
1.3.6.1.4.1.897.2.4.1.2.1.3.db	Integer32	saSrvPropType	2[2]
1.3.6.1.4.1.897.2.4.1.2.1.4.db	OID	saSrvPropOID	OID of SQL Anywhere MIB entry[3]
1.3.6.1.4.1.897.2.4.1.2.1.5.db	String	saSrvPropName	Property name

[1] Values: 1=Server, 2=Database

[2] Values: 1=Statistic, 2=Property, 3=Option

[3] The OID returned does not include the database number. You must append the database number to the OID before it can be used in a query.

1.8.2.3.8.2.3 saDbMetaData.saDbStatMetaDataTable

Contains metadata about the database statistics.

The value db is the database number in the `sasnmp.ini` file.

OID	Type	Name	Value returned
1.3.6.1.4.1.897.2.4.2.1.1.1.db	Integer32	saDbStatIndex	db
1.3.6.1.4.1.897.2.4.2.1.1.2.db	Integer32	saDbStatObjType	2[1]
1.3.6.1.4.1.897.2.4.2.1.1.3.db	Integer32	saDbStatType	1[2]
1.3.6.1.4.1.897.2.4.2.1.1.4.db	OID	saDbStatOID	OID of SQL Anywhere MIB entry[3]
1.3.6.1.4.1.897.2.4.2.1.1.5.db	String	saDbStatName	Statistic name

[1] Values: 1=Server, 2=Database

[2] Values: 1=Statistic, 2=Property, 3=Option

[3] The OID returned does not include the database number. You must append the database number to the OID before it can be used in a query.

1.8.2.3.8.2.4 saDbMetaData.saDbPropMetaDataTable

Contains metadata about the database properties.

The value db is the database number in the `sasnmp.ini` file.

OID	Type	Name	Value returned
1.3.6.1.4.1.897.2.4.2.2.1.1.db	Integer32	saDbPropIndex	db
1.3.6.1.4.1.897.2.4.2.2.1.2.db	Integer32	saDbPropObjType	2[1]
1.3.6.1.4.1.897.2.4.2.2.1.3.db	Integer32	saDbPropType	2[2]
1.3.6.1.4.1.897.2.4.2.2.1.4.db	OID	saDbPropOID	OID of SQL Anywhere MIB entry[3]
1.3.6.1.4.1.897.2.4.2.2.1.5.db	String	saDbPropName	Property name

[1] Values: 1=Server, 2=Database

[2] Values: 1=Statistic, 2=Property, 3=Option

[3] The OID returned does not include the database number. You must append the database number to the OID before it can be used in a query.

1.8.2.3.8.2.5 saDbMetaData.saDbOptMetaDataTable

Contains metadata about the database options.

The value db is the database number in the `sasnmp.ini` file.

OID	Type	Name	Value returned
1.3.6.1.4.1.897.2.4.2.1.1.1.db	Integer32	saDbOptIndex	db
1.3.6.1.4.1.897.2.4.2.1.1.2.db	Integer32	saDbOptObjType	2[1]
1.3.6.1.4.1.897.2.4.2.1.1.3.db	Integer32	saDbOptType	3[2]
1.3.6.1.4.1.897.2.4.2.1.1.4.db	OID	saDbOptOID	OID of SQL Anywhere MIB entry[3]
1.3.6.1.4.1.897.2.4.2.1.1.5.db	String	saDbOptName	Option name

[1] Values: 1=Server, 2=Database

[2] Values: 1=Statistic, 2=Property, 3=Option

[3] The OID returned does not include the database number. You must append the database number to the OID before it can be used in a query.

1.8.2.3.8.3 SQL Anywhere MIB Database Server Statistics

Retrieve database server statistics by using the SQL Anywhere SNMP Extension Agent.

Writable options are marked with an asterisk (*). The value `n` is the database number in the `sasnmp.ini` file.

Descriptions

OID	Type	Name	Statistic
1.3.6.1.4.1.897.2.1.1.1.n	Integer32	srvStatActiveReq	ActiveReq
1.3.6.1.4.1.897.2.1.1.2.n	Integer32	srvStatAvailIO	AvailIO
1.3.6.1.4.1.897.2.1.1.3.n	Counter64	srvStatBytesReceived	BytesReceived
1.3.6.1.4.1.897.2.1.1.4.n	Counter64	srvStatBytesReceivedUncomp	BytesReceivedUncomp
1.3.6.1.4.1.897.2.1.1.5.n	Counter64	srvStatBytesSent	BytesSent
1.3.6.1.4.1.897.2.1.1.6.n	Counter64	srvStatBytesSentUncomp	BytesSentUncomp
1.3.6.1.4.1.897.2.1.1.7.n	Counter64	srvStatCacheHits	CacheHits
1.3.6.1.4.1.897.2.1.1.8.n	Integer32	srvStatCachePinned	CachePinned
1.3.6.1.4.1.897.2.1.1.9.n	Counter64	srvStatCacheRead	CacheRead
1.3.6.1.4.1.897.2.1.1.10.n	Counter64	srvStatCacheReplacements	CacheReplacements
1.3.6.1.4.1.897.2.1.1.11.n	Integer32	srvStatCurrentCacheSize	CurrentCacheSize
1.3.6.1.4.1.897.2.1.1.12.n	Counter64	srvStatDiskRead	DiskRead

OID	Type	Name	Statistic
1.3.6.1.4.1.897.2.1.1.13.n	Integer32	srvStatFreeBuffers	FreeBuffers
1.3.6.1.4.1.897.2.1.1.14.n	Integer32	srvStatInternal	Internal
1.3.6.1.4.1.897.2.1.1.15.n	Integer32	srvStatUniqueClientAddresses	UniqueClientAddresses
1.3.6.1.4.1.897.2.1.1.16.n	Integer32	srvStatLockedHeapPages	LockedHeapPages
1.3.6.1.4.1.897.2.1.1.17.n	Counter64	srvStatMainHeapBytes	MainHeapBytes
1.3.6.1.4.1.897.2.1.1.18.n	Integer32	srvStatMainHeapPages	MainHeapPages
1.3.6.1.4.1.897.2.1.1.19.n	Integer32	srvStatMapPhysicalMemoryEng	MapPhysicalMemoryEng
1.3.6.1.4.1.897.2.1.1.20.n	Integer32	srvStatMaxCacheSize	MaxCacheSize
1.3.6.1.4.1.897.2.1.1.21.n	Integer32	srvStatMinCacheSize	MinCacheSize
1.3.6.1.4.1.897.2.1.1.22.n	Counter64	srvStatMultiPacketsReceived	MultiPacketsReceived
1.3.6.1.4.1.897.2.1.1.23.n	Counter64	srvStatMultiPacketsSent	MultiPacketsSent
1.3.6.1.4.1.897.2.1.1.24.n	Counter64	srvStatPacketsReceived	PacketsReceived
1.3.6.1.4.1.897.2.1.1.25.n	Counter64	srvStatPacketsReceivedUncomp	PacketsReceivedUncomp
1.3.6.1.4.1.897.2.1.1.26.n	Counter64	srvStatPacketsSent	PacketsSent
1.3.6.1.4.1.897.2.1.1.27.n	Counter64	srvStatPacketsSentUncomp	PacketsSentUncomp
1.3.6.1.4.1.897.2.1.1.28.n	Integer32	srvStatPeakCacheSize	PeakCacheSize
1.3.6.1.4.1.897.2.1.1.29.n	Integer32	srvStatRemoteputWait	RemoteputWait
1.3.6.1.4.1.897.2.1.1.30.n	Integer32	srvStatReq	Req
1.3.6.1.4.1.897.2.1.1.31.n	Counter64	srvStatSendFail	SendFail
1.3.6.1.4.1.897.2.1.1.32.n	Integer32	srvStatTotalBuffers	TotalBuffers
1.3.6.1.4.1.897.2.1.1.33.n	Integer32	srvStatUnschReq	UnschReq
1.3.6.1.4.1.897.2.1.1.34.n	Integer32	srvStatInternal	Internal
1.3.6.1.4.1.897.2.1.1.35.n	Integer32	srvStatCacheFile	CacheFile
1.3.6.1.4.1.897.2.1.1.36.n	Integer32	srvStatCacheFileDirty	CacheFileDirty
1.3.6.1.4.1.897.2.1.1.37.n	Integer32	srvStatCacheAllocated	CacheAllocated
1.3.6.1.4.1.897.2.1.1.38.n	Integer32	srvStatCachePanics	CachePanics
1.3.6.1.4.1.897.2.1.1.39.n	Integer32	srvStatCacheFree	CacheFree
1.3.6.1.4.1.897.2.1.1.40.n	Integer32	srvStatCacheScavenges	CacheScavenges
1.3.6.1.4.1.897.2.1.1.41.n	Integer32	srvStatCacheScavengeVisited	CacheScavengeVisited
1.3.6.1.4.1.897.2.1.1.42.n	Integer32	srvStatLockedCursorPages	LockedCursorPages
1.3.6.1.4.1.897.2.1.1.43.n	Integer32	srvStatQueryHeapPages	QueryHeapPages

OID	Type	Name	Statistic
1.3.6.1.4.1.897.2.1.1.44.n	Integer32	srvStatCarverHeapPages	CarverHeapPages
1.3.6.1.4.1.897.2.1.1.45.n	Integer32	srvStatHeapsRelocatable	HeapsRelocatable
1.3.6.1.4.1.897.2.1.1.46.n	Integer32	srvStatHeapsLocked	HeapsLocked
1.3.6.1.4.1.897.2.1.1.47.n	Integer32	srvStatHeapsQuery	HeapsQuery
1.3.6.1.4.1.897.2.1.1.48.n	Integer32	srvStatHeapsCarver	HeapsCarver
1.3.6.1.4.1.897.2.1.1.49.n	Integer32	srvStatMultiPageAllocs	MultiPageAllocs
1.3.6.1.4.1.897.2.1.1.50.n	Integer32	srvStatRequestsReceived	RequestsReceived
1.3.6.1.4.1.897.2.1.1.51.n	Integer32	srvStatExchangeTasks	ExchangeTasks
1.3.6.1.4.1.897.2.1.1.52.n	Integer32	srvStatClientStmtCacheHits	ClientStmtCacheHits
1.3.6.1.4.1.897.2.1.1.53.n	Integer32	srvStatClientStmtCache-Misses	ClientStmtCacheMisses
1.3.6.1.4.1.897.2.1.1.54.n	Integer32	srvStatQueryMemActiveCurr	QueryMemActiveCurr
1.3.6.1.4.1.897.2.1.1.55.n	Integer32	srvStatQueryMemActiveEst	QueryMemActiveEst
1.3.6.1.4.1.897.2.1.1.56.n	Integer32	srvStatQueryMemGrantWait-ing	QueryMemGrantWaiting
1.3.6.1.4.1.897.2.1.1.57.n	Integer32	srvStatQueryMemGrantRe-quested	QueryMemGrantRequested
1.3.6.1.4.1.897.2.1.1.58.n	Integer32	srvStatQueryMem-GrantWaited	QueryMemGrantWaited
1.3.6.1.4.1.897.2.1.1.59.n	Integer32	srvStatQueryMemGrant-Failed	QueryMemGrantFailed
1.3.6.1.4.1.897.2.1.1.60.n	Integer32	srvStatQueryMemGrant-Granted	QueryMemGrantGranted
1.3.6.1.4.1.897.2.1.1.61.n	Integer32	srvStatQueryMemExtraAvail	QueryMemExtraAvail
1.3.6.1.4.1.897.2.1.1.62.n	Integer32	srvStatThreadDeadlocksA-voided	ThreadDeadlocksAvoided
1.3.6.1.4.1.897.2.1.1.63.n	Integer32	srvStatThreadDeadlocksRe-ported	ThreadDeadlocksReported
1.3.6.1.4.1.897.2.1.1.64.n	Integer32	srvStatCommit	Commit
1.3.6.1.4.1.897.2.1.1.65.n	Integer32	srvStatCursor	Cursor
1.3.6.1.4.1.897.2.1.1.66.n	Integer32	srvStatCursorOpen	CursorOpen
1.3.6.1.4.1.897.2.1.1.67.n	Integer32	srvStatRlbk	Rlbk
1.3.6.1.4.1.897.2.1.1.68.n	Integer32	srvStatPrepStmt	PrepStmt
1.3.6.1.4.1.897.2.1.1.69.n	Integer32	srvStatParameterizationPre-pareCount	ParameterizationPrepare-Count
1.3.6.1.4.1.897.2.1.1.70.n	Integer32	srvStatConnCount	ConnCount
1.3.6.1.4.1.897.2.1.1.71.n	Integer32	srvStatCompletedReq	CompletedReq

OID	Type	Name	Statistic
1.3.6.1.4.1.897.2.1.1.72.n	Integer32	srvStatCurrRead	CurrRead
1.3.6.1.4.1.897.2.1.1.73.n	Integer32	srvStatCurrWrite	CurrWrite
1.3.6.1.4.1.897.2.1.1.74.n	Counter64	srvStatDiskWrite	DiskWrite
1.3.6.1.4.1.897.2.1.1.75.n	Integer32	srvStatCacheMisses	CacheMisses

Related Information

List of Database Server Properties [page 900]
List of Database Server Properties [page 900]

1.8.2.3.8.4 SQL Anywhere MIB Database Server Properties

Retrieve database server properties by using the SQL Anywhere SNMP Extension Agent.

Writable options are marked with an asterisk (*). The value `n` is the database number in the `sasnmp.ini` file.

Descriptions

OID	Type	Name	Property
1.3.6.1.4.1.897.2.1.2.1.n	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.2.n	String	srvPropCharSet	CharSet
1.3.6.1.4.1.897.2.1.2.3.n	String	srvPropCommandLine	CommandLine
1.3.6.1.4.1.897.2.1.2.4.n	String	srvPropCompactPlatformVer	CompactPlatformVer
1.3.6.1.4.1.897.2.1.2.5.n	String	srvPropCompanyName	CompanyName
1.3.6.1.4.1.897.2.1.2.6.n	String	srvPropConnsDisabled*	ConnsDisabled
1.3.6.1.4.1.897.2.1.2.7.n	String	srvPropConsoleLogFile	ConsoleLogFile
1.3.6.1.4.1.897.2.1.2.8.n	String	srvPropDefaultCollation	DefaultCollation
1.3.6.1.4.1.897.2.1.2.9.n	String	srvPropIdleTimeout	IdleTimeout
1.3.6.1.4.1.897.2.1.2.10.n	String	srvPropIsIQ	IsIQ
1.3.6.1.4.1.897.2.1.2.11.n	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.12.n	String	srvPropIsNetworkServer	IsNetworkServer
1.3.6.1.4.1.897.2.1.2.13.n	String	srvPropIsRuntimeServer	IsRuntimeServer

OID	Type	Name	Property
1.3.6.1.4.1.897.2.1.2.14.n	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.15.n	String	srvPropLanguage	Language
1.3.6.1.4.1.897.2.1.2.16.n	String	srvPropLegalCopyright	LegalCopyright
1.3.6.1.4.1.897.2.1.2.17.n	String	srvPropLegalTrademarks	LegalTrademarks
1.3.6.1.4.1.897.2.1.2.18.n	String	srvPropLicenseCount	LicenseCount
1.3.6.1.4.1.897.2.1.2.19.n	String	srvPropLicensedCompany	LicensedCompany
1.3.6.1.4.1.897.2.1.2.20.n	String	srvPropLicensedUser	LicensedUser
1.3.6.1.4.1.897.2.1.2.21.n	String	srvPropLicenseType	LicenseType
1.3.6.1.4.1.897.2.1.2.22.n	String	srvPropLivenessTimeout*	LivenessTimeout
1.3.6.1.4.1.897.2.1.2.23.n	String	srvPropMachineName	MachineName
1.3.6.1.4.1.897.2.1.2.24.n	String	srvPropMaxMessage	MaxMessage
1.3.6.1.4.1.897.2.1.2.25.n	String	srvPropMessageWindowSize	MessageWindowSize
1.3.6.1.4.1.897.2.1.2.26.n	String	srvPropName	Name
1.3.6.1.4.1.897.2.1.2.27.n	String	srvPropNativeProcessor-Architecture	NativeProcessorArchitecture
1.3.6.1.4.1.897.2.1.2.28.n	String	srvPropNumPhysicalProces-sors	NumPhysicalProcessors
1.3.6.1.4.1.897.2.1.2.29.n	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.30.n	String	srvPropOmniIdentifier	OmniIdentifier
1.3.6.1.4.1.897.2.1.2.31.n	String	srvPropPageSize	PageSize
1.3.6.1.4.1.897.2.1.2.32.n	String	srvPropPlatform	Platform
1.3.6.1.4.1.897.2.1.2.33.n	String	srvPropPlatformVer	PlatformVer
1.3.6.1.4.1.897.2.1.2.34.n	String	srvPropProcessCPU	ProcessCPU
1.3.6.1.4.1.897.2.1.2.35.n	String	srvPropProcessCPUSystem	ProcessCPUSystem
1.3.6.1.4.1.897.2.1.2.36.n	String	srvPropProcessCPUUser	ProcessCPUUser
1.3.6.1.4.1.897.2.1.2.37.n	String	srvPropProcessorArchitec-ture	ProcessorArchitecture
1.3.6.1.4.1.897.2.1.2.38.n	String	srvPropProductName	ProductName
1.3.6.1.4.1.897.2.1.2.39.n	String	srvPropProductVersion	ProductVersion
1.3.6.1.4.1.897.2.1.2.40.n	String	srvPropQuittingTime*	QuittingTime
1.3.6.1.4.1.897.2.1.2.41.n	String	srvPropRememberLast-Statement*	RememberLastStatement
1.3.6.1.4.1.897.2.1.2.42.n	String	srvPropRequestFilterConn	RequestFilterConn
1.3.6.1.4.1.897.2.1.2.43.n	String	srvPropRequestFilterDB	RequestFilterDB
1.3.6.1.4.1.897.2.1.2.44.n	String	srvPropRequestLogFile*	RequestLogFile

OID	Type	Name	Property
1.3.6.1.4.1.897.2.1.2.45.n	String	srvPropRequestLogging*	RequestLogging
1.3.6.1.4.1.897.2.1.2.46.n	String	srvPropRequestLogMaxSize	RequestLogMaxSize
1.3.6.1.4.1.897.2.1.2.47.n	String	srvPropStartTime	StartTime
1.3.6.1.4.1.897.2.1.2.48.n	String	srvPropTempDir	TempDir
1.3.6.1.4.1.897.2.1.2.49.n	String	srvPropMultiProgrammin-gLevel	MultiProgrammingLevel
1.3.6.1.4.1.897.2.1.2.50.n	String	srvPropTimeZoneAdjust-ment	TimeZoneAdjustment
1.3.6.1.4.1.897.2.1.2.51.n	String	srvPropHttpPorts	HttpPorts
1.3.6.1.4.1.897.2.1.2.52.n	String	srvPropHttpsPorts	HttpsPorts
1.3.6.1.4.1.897.2.1.2.53.n	String	srvPropProfileFilterConn	ProfileFilterConn
1.3.6.1.4.1.897.2.1.2.54.n	String	srvPropProfileFilterUser	ProfileFilterUser
1.3.6.1.4.1.897.2.1.2.55.n	String	srvPropRequestLogNumFiles	RequestLogNumFiles
1.3.6.1.4.1.897.2.1.2.56.n	String	srvPropIsFipsAvailable	IsFipsAvailable
1.3.6.1.4.1.897.2.1.2.57.n	String	srvPropFipsMode	FipsMode
1.3.6.1.4.1.897.2.1.2.58.n	String	srvPropStartDBPermission	StartDBPermission
1.3.6.1.4.1.897.2.1.2.59.n	String	srvPropServerName	ServerName
1.3.6.1.4.1.897.2.1.2.60.n	String	srvPropRememberLastPlan	RememberLastPlan
1.3.6.1.4.1.897.2.1.2.61.n	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.62.n	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.63.n	String	srvPropRequestTiming	RequestTiming
1.3.6.1.4.1.897.2.1.2.64.n	String	srvPropCacheSizingStatis-tics	CacheSizingStatistics
1.3.6.1.4.1.897.2.1.2.65.n	String	srvPropConsoleLogMaxSize	ConsoleLogMaxSize
1.3.6.1.4.1.897.2.1.2.66.n	String	srvPropDebuggingInforma-tion	DebuggingInformation
1.3.6.1.4.1.897.2.1.2.67.n	String	srvPropMessage	Message
1.3.6.1.4.1.897.2.1.2.68.n	String	srvPropMessageText	MessageText
1.3.6.1.4.1.897.2.1.2.69.n	String	srvPropMessageTime	MessageTime
1.3.6.1.4.1.897.2.1.2.70.n	String	srvPropIsRsaAvailable	IsRsaAvailable
1.3.6.1.4.1.897.2.1.2.71.n	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.72.n	String	srvPropMaxConnections	MaxConnections
1.3.6.1.4.1.897.2.1.2.73.n	String	srvPropNumLogicalProces-sors	NumLogicalProcessors
1.3.6.1.4.1.897.2.1.2.74.n	String	srvPropNumLogicalProces-sorsUsed	NumLogicalProcessorsUsed

OID	Type	Name	Property
1.3.6.1.4.1.897.2.1.2.75.n	String	srvPropNumPhysicalProcessorsUsed	NumPhysicalProcessorsUsed
1.3.6.1.4.1.897.2.1.2.76.n	String	srvPropDefaultNcharCollation	DefaultNcharCollation
1.3.6.1.4.1.897.2.1.2.77.n	String	srvPropCollectStatistics	CollectStatistics
1.3.6.1.4.1.897.2.1.2.78.n	String	srvPropFirstOption	FirstOption
1.3.6.1.4.1.897.2.1.2.79.n	String	srvPropLastOption	LastOption
1.3.6.1.4.1.897.2.1.2.80.n	String	srvPropLastConnectionProperty	LastConnectionProperty
1.3.6.1.4.1.897.2.1.2.81.n	String	srvPropLastDatabaseProperty	LastDatabaseProperty
1.3.6.1.4.1.897.2.1.2.82.n	String	srvPropLastServerProperty	LastServerProperty
1.3.6.1.4.1.897.2.1.2.83.n	String	srvPropWebClientLogging	WebClientLogging
1.3.6.1.4.1.897.2.1.2.84.n	String	srvPropWebClientLogFile	WebClientLogFile
1.3.6.1.4.1.897.2.1.2.85.n	String	srvPropHttpNumConnections	HttpNumConnections
1.3.6.1.4.1.897.2.1.2.86.n	String	srvPropHttpsNumConnections	HttpsNumConnections
1.3.6.1.4.1.897.2.1.2.87.n	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.88.n	String	srvPropHttpNumActiveReq	HttpNumActiveReq
1.3.6.1.4.1.897.2.1.2.89.n	String	srvPropHttpsNumActiveReq	HttpsNumActiveReq
1.3.6.1.4.1.897.2.1.2.90.n	String	srvPropHttpNumSessions	HttpNumSessions
1.3.6.1.4.1.897.2.1.2.91.n	String	srvPropQueryMemPages	QueryMemPages
1.3.6.1.4.1.897.2.1.2.92.n	String	srvPropQueryMemGrantBase	QueryMemGrantBase
1.3.6.1.4.1.897.2.1.2.93.n	String	srvPropQueryMemGrantBaseMI	QueryMemGrantBaseMI
1.3.6.1.4.1.897.2.1.2.94.n	String	srvPropQueryMemGrantExtra	QueryMemGrantExtra
1.3.6.1.4.1.897.2.1.2.95.n	String	srvPropQueryMemActiveMax	QueryMemActiveMax
1.3.6.1.4.1.897.2.1.2.96.n	String	srvPropQueryMemPercentOfCache	QueryMemPercentOfCache
1.3.6.1.4.1.897.2.1.2.97.n	String	srvPropMessageCategoryLimit	MessageCategoryLimit
1.3.6.1.4.1.897.2.1.2.98.n	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.99.n	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.100.n	String	srvPropIsService	IsService

OID	Type	Name	Property
1.3.6.1.4.1.897.2.1.2.101.n	String	srvPropTcpIpAddresses	TcpIpAddresses
1.3.6.1.4.1.897.2.1.2.102.n	String	srvPropHttpAddresses	HttpAddresses
1.3.6.1.4.1.897.2.1.2.103.n	String	srvPropHttpsAddresses	HttpsAddresses
1.3.6.1.4.1.897.2.1.2.104.n	String	srvPropProcessID	ProcessID
1.3.6.1.4.1.897.2.1.2.105.n	String	srvPropRemoteCapability	RemoteCapability
1.3.6.1.4.1.897.2.1.2.106.n	String	srvPropMaxRemoteCapability	MaxRemoteCapability
1.3.6.1.4.1.897.2.1.2.107.n	String	srvPropEventTypeName	EventTypeName
1.3.6.1.4.1.897.2.1.2.108.n	String	srvPropEventTypeDesc	EventTypeDesc
1.3.6.1.4.1.897.2.1.2.109.n	String	srvPropMaxEventType	MaxEventType
1.3.6.1.4.1.897.2.1.2.110.n	String	srvPropIsPortableDevice	IsPortableDevice
1.3.6.1.4.1.897.2.1.2.111.n	String	srvPropIPAddressMonitorPeriod	IPAddressMonitorPeriod
1.3.6.1.4.1.897.2.1.2.112.n	String	srvPropServerEdition	ServerEdition
1.3.6.1.4.1.897.2.1.2.113.n	String	srvPropObjectType	ObjectType
1.3.6.1.4.1.897.2.1.2.114.n	String	srvPropCurrentMultiProgrammingLevel	CurrentMultiProgrammingLevel
1.3.6.1.4.1.897.2.1.2.115.n	String	srvPropMinMultiProgrammingLevel	MinMultiProgrammingLevel
1.3.6.1.4.1.897.2.1.2.116.n	String	srvPropMaxMultiProgrammingLevel	MaxMultiProgrammingLevel
1.3.6.1.4.1.897.2.1.2.117.n	String	srvPropAutoMultiProgrammingLevel	AutoMultiProgrammingLevel
1.3.6.1.4.1.897.2.1.2.118.n	String	srvPropAutoMultiProgrammingLevelStatistics	AutoMultiProgrammingLevelStatistics
1.3.6.1.4.1.897.2.1.2.119.n	String	srvPropApproximateCPUTime	ApproximateCPUTime
1.3.6.1.4.1.897.2.1.2.120.n	String	srvPropConnectedTime	ConnectedTime
1.3.6.1.4.1.897.2.1.2.121.n	String	srvPropReqCountUnscheduled	ReqCountUnscheduled
1.3.6.1.4.1.897.2.1.2.122.n	String	srvPropReqCountActive	ReqCountActive
1.3.6.1.4.1.897.2.1.2.123.n	String	srvPropReqCountBlockIO	ReqCountBlockIO
1.3.6.1.4.1.897.2.1.2.124.n	String	srvPropReqCountBlockLock	ReqCountBlockLock
1.3.6.1.4.1.897.2.1.2.125.n	String	srvPropReqCountBlockContention	ReqCountBlockContention
1.3.6.1.4.1.897.2.1.2.126.n	String	srvPropReqTimeUnscheduled	ReqTimeUnscheduled
1.3.6.1.4.1.897.2.1.2.127.n	String	srvPropReqTimeActive	ReqTimeActive

OID	Type	Name	Property
1.3.6.1.4.1.897.2.1.2.128.n	String	srvPropReqTimeBlockIO	ReqTimeBlockIO
1.3.6.1.4.1.897.2.1.2.129.n	String	srvPropReqTimeBlockLock	ReqTimeBlockLock
1.3.6.1.4.1.897.2.1.2.130.n	String	srvPropReqTimeBlockContention	ReqTimeBlockContention
1.3.6.1.4.1.897.2.1.2.131.n	String	srvPropLicenseKey	LicenseKey
1.3.6.1.4.1.897.2.1.2.132.n	String	srvPropDiskSandbox	DiskSandbox
1.3.6.1.4.1.897.2.1.2.133.n	String	srvPropProcessorAffinity	ProcessorAffinity
1.3.6.1.4.1.897.2.1.2.134.n	String	srvPropIsAesniAvailable	IsAesniAvailable
1.3.6.1.4.1.897.2.1.2.135.n	String	srvPropCurrentMirrorBackgroundWorkers	CurrentMirrorBackgroundWorkers
1.3.6.1.4.1.897.2.1.2.136.n	String	srvPropMaxMirrorBackgroundWorkers	MaxMirrorBackgroundWorkers
1.3.6.1.4.1.897.2.1.2.137.n	String	srvPropHasSecuredFeature	HasSecuredFeature
1.3.6.1.4.1.897.2.1.2.138.n	String	srvPropHasSecureFeatureKey	HasSecureFeatureKey
1.3.6.1.4.1.897.2.1.2.139.n	String	srvPropTcpIpListeners	TcpIpListeners
1.3.6.1.4.1.897.2.1.2.140.n	String	srvPropHttpListeners	HttpListeners
1.3.6.1.4.1.897.2.1.2.141.n	String	srvPropHttpsListeners	HttpsListeners
1.3.6.1.4.1.897.2.1.2.142.n	String	srvPropODataAddresses	ODataAddresses
1.3.6.1.4.1.897.2.1.2.143.n	String	srvPropODataSecureAddresses	ODataSecureAddresses
1.3.6.1.4.1.897.2.1.2.144.n	String	srvPropTopologyAwareScheduling	TopologyAwareScheduling
1.3.6.1.4.1.897.2.1.2.145.n	String	srvPropSharedMemoryListener	SharedMemoryListener
1.3.6.1.4.1.897.2.1.2.146.n	String	srvPropCockpitDB	CockpitDB
1.3.6.1.4.1.897.2.1.2.147.n	String	srvPropSingleJVM	SingleJVM
1.3.6.1.4.1.897.2.1.2.148.n	String	srvPropSingleCLR	SingleCLR
1.3.6.1.4.1.897.2.1.2.149.n	String	srvPropJavaVM	JavaVM
1.3.6.1.4.1.897.2.1.2.150.n	String	srvPropPlanStatisticsStored	PlanStatisticsStored
1.3.6.1.4.1.897.2.1.2.151.n	String	srvPropPlanStatisticsStoredMax	PlanStatisticsStoredMax
1.3.6.1.4.1.897.2.1.2.152.n	String	srvPropPropertyHistoryList	PropertyHistoryList
1.3.6.1.4.1.897.2.1.2.153.n	String	srvPropPropertyHistoryListActual	PropertyHistoryListActual
1.3.6.1.4.1.897.2.1.2.154.n	String	srvPropPropertyHistorySize	PropertyHistorySize
1.3.6.1.4.1.897.2.1.2.155.n	String	srvPropPropertyHistorySizeBytes	PropertyHistorySizeBytes

OID	Type	Name	Property
1.3.6.1.4.1.897.2.1.2.156.n	String	srvPropCockpitURL	CockpitURL

Related Information

List of Database Server Properties [page 900]

1.8.2.3.8.5 SQL Anywhere MIB Database Statistics

Retrieve database statistics by using the SQL Anywhere SNMP Extension Agent.

Writable options are marked with an asterisk (*). The value `n` is the database number in the `sasnmp.ini` file.

Descriptions

OID	Type	Name	Statistic
1.3.6.1.4.1.897.2.2.1.1.n	Counter64	dbStatCacheHits	CacheHits
1.3.6.1.4.1.897.2.2.1.2.n	Integer32	dbStatCacheReadIndInt	CacheReadIndInt
1.3.6.1.4.1.897.2.2.1.3.n	Integer32	dbStatCacheReadIndLeaf	CacheReadIndLeaf
1.3.6.1.4.1.897.2.2.1.4.n	Counter64	dbStatCacheRead	CacheRead
1.3.6.1.4.1.897.2.2.1.5.n	Integer32	dbStatCacheReadTable	CacheReadTable
1.3.6.1.4.1.897.2.2.1.6.n	Integer32	dbStatChkpt	Chkpt
1.3.6.1.4.1.897.2.2.1.7.n	Integer32	dbStatChkptFlush	ChkptFlush
1.3.6.1.4.1.897.2.2.1.8.n	Integer32	dbStatChkptPage	ChkptPage
1.3.6.1.4.1.897.2.2.1.9.n	Integer32	dbStatCheckpointUrgency	CheckpointUrgency
1.3.6.1.4.1.897.2.2.1.10.n	Integer32	dbStatInternal	Internal
1.3.6.1.4.1.897.2.2.1.11.n	Integer32	dbStatInternal	Internal
1.3.6.1.4.1.897.2.2.1.12.n	Integer32	dbStatCheckpointLogCommitToDisk	CheckpointLogCommitToDisk
1.3.6.1.4.1.897.2.2.1.13.n	Integer32	dbStatCheckpointLogPagesInUse	CheckpointLogPagesInUse
1.3.6.1.4.1.897.2.2.1.14.n	Integer32	dbStatCheckpointLogPagesRelocated	CheckpointLogPagesRelocated
1.3.6.1.4.1.897.2.2.1.15.n	Integer32	dbStatCheckpointLogSavePreimage	CheckpointLogSavePreimage

OID	Type	Name	Statistic
1.3.6.1.4.1.897.2.2.1.16.n	Integer32	dbStatCheckpointLogSize	CheckpointLogSize
1.3.6.1.4.1.897.2.2.1.17.n	Integer32	dbStatCheckpointLogWrites	CheckpointLogWrites
1.3.6.1.4.1.897.2.2.1.18.n	Integer32	dbStatCheckpointLogPages-Written	CheckpointLogPagesWritten
1.3.6.1.4.1.897.2.2.1.19.n	Integer32	dbStatCommitFile	CommitFile
1.3.6.1.4.1.897.2.2.1.20.n	Integer32	dbStatConnCount	ConnCount
1.3.6.1.4.1.897.2.2.1.21.n	Integer32	dbStatCurrIO	CurrIO
1.3.6.1.4.1.897.2.2.1.22.n	Integer32	dbStatCurrRead	CurrRead
1.3.6.1.4.1.897.2.2.1.23.n	Integer32	dbStatCurrWrite	CurrWrite
1.3.6.1.4.1.897.2.2.1.24.n	Integer32	dbStatDiskReadIndInt	DiskReadIndInt
1.3.6.1.4.1.897.2.2.1.25.n	Integer32	dbStatDiskReadIndLeaf	DiskReadIndLeaf
1.3.6.1.4.1.897.2.2.1.26.n	Counter64	dbStatDiskRead	DiskRead
1.3.6.1.4.1.897.2.2.1.27.n	Integer32	dbStatDiskReadTable	DiskReadTable
1.3.6.1.4.1.897.2.2.1.28.n	Counter64	dbStatDiskWrite	DiskWrite
1.3.6.1.4.1.897.2.2.1.29.n	Integer32	dbStatExtendDB	ExtendDB
1.3.6.1.4.1.897.2.2.1.30.n	Integer32	dbStatExtendTempWrite	ExtendTempWrite
1.3.6.1.4.1.897.2.2.1.31.n	Integer32	dbStatFullCompare	FullCompare
1.3.6.1.4.1.897.2.2.1.32.n	Integer32	dbStatGetData	GetData
1.3.6.1.4.1.897.2.2.1.33.n	Integer32	dbStatIdleCheck	IdleCheck
1.3.6.1.4.1.897.2.2.1.34.n	Integer32	dbStatIdleChkpt	IdleChkpt
1.3.6.1.4.1.897.2.2.1.35.n	Integer32	dbStatIdleChkTime	IdleChkTime
1.3.6.1.4.1.897.2.2.1.36.n	Integer32	dbStatIdleWrite	IdleWrite
1.3.6.1.4.1.897.2.2.1.37.n	Integer32	dbStatIndAdd	IndAdd
1.3.6.1.4.1.897.2.2.1.38.n	Integer32	dbStatIndLookup	IndLookup
1.3.6.1.4.1.897.2.2.1.39.n	Integer32	dbStatIOToRecover	IOToRecover
1.3.6.1.4.1.897.2.2.1.40.n	Integer32	dbStatInternal	Internal
1.3.6.1.4.1.897.2.2.1.41.n	Integer32	dbStatInternal	Internal
1.3.6.1.4.1.897.2.2.1.42.n	Integer32	dbStatLockTablePages	LockTablePages
1.3.6.1.4.1.897.2.2.1.43.n	Integer32	dbStatInternal	Internal
1.3.6.1.4.1.897.2.2.1.44.n	Integer32	dbStatMaxIO	MaxIO
1.3.6.1.4.1.897.2.2.1.45.n	Integer32	dbStatMaxRead	MaxRead
1.3.6.1.4.1.897.2.2.1.46.n	Integer32	dbStatMaxWrite	MaxWrite
1.3.6.1.4.1.897.2.2.1.47.n	Counter64	dbStatPageRelocations	PageRelocations
1.3.6.1.4.1.897.2.2.1.48.n	Integer32	dbStatProcedurePages	ProcedurePages

OID	Type	Name	Statistic
1.3.6.1.4.1.897.2.2.1.49.n	Integer32	dbStatQueryCachePages	QueryCachePages
1.3.6.1.4.1.897.2.2.1.50.n	Integer32	dbStatQueryLowMemory-Strategy	QueryLowMemoryStrategy
1.3.6.1.4.1.897.2.2.1.51.n	Counter64	dbStatQueryRowsMaterial-ized	QueryRowsMaterialized
1.3.6.1.4.1.897.2.2.1.52.n	Integer32	dbStatRecoveryUrgency	RecoveryUrgency
1.3.6.1.4.1.897.2.2.1.53.n	Integer32	dbStatLogFreeCommit	LogFreeCommit
1.3.6.1.4.1.897.2.2.1.54.n	Integer32	dbStatLogWrite	LogWrite
1.3.6.1.4.1.897.2.2.1.55.n	Integer32	dbStatRelocatableHeap-Pages	RelocatableHeapPages
1.3.6.1.4.1.897.2.2.1.56.n	Integer32	dbStatRollbackLogPages	RollbackLogPages
1.3.6.1.4.1.897.2.2.1.57.n	Integer32	dbStatTempTablePages	TempTablePages
1.3.6.1.4.1.897.2.2.1.58.n	Integer32	dbStatTriggerPages	TriggerPages
1.3.6.1.4.1.897.2.2.1.59.n	Integer32	dbStatViewPages	ViewPages
1.3.6.1.4.1.897.2.2.1.60.n	Integer32	dbStatVersionStorePages	VersionStorePages
1.3.6.1.4.1.897.2.2.1.61.n	Integer32	dbStatSnapshotCount	SnapshotCount
1.3.6.1.4.1.897.2.2.1.62.n	Integer32	dbStatLockCount	LockCount
1.3.6.1.4.1.897.2.2.1.63.n	Integer32	dbStatCacheReadWorkTable	CacheReadWorkTable
1.3.6.1.4.1.897.2.2.1.64.n	Integer32	dbStatDiskReadWorkTable	DiskReadWorkTable
1.3.6.1.4.1.897.2.2.1.65.n	Integer32	dbStatPrepares	Prepares
1.3.6.1.4.1.897.2.2.1.66.n	Integer32	dbStatConnPoolCached-Count	ConnPoolCachedCount
1.3.6.1.4.1.897.2.2.1.67.n	Integer32	dbStatConnPoolHits	ConnPoolHits
1.3.6.1.4.1.897.2.2.1.68.n	Integer32	dbStatConnPoolMisses	ConnPoolMisses
1.3.6.1.4.1.897.2.2.1.69.n	Integer32	dbStatHttpConnPoolCa-chedCount	HttpConnPoolCachedCount
1.3.6.1.4.1.897.2.2.1.70.n	Integer32	dbStatHttpConnPoolHits	HttpConnPoolHits
1.3.6.1.4.1.897.2.2.1.71.n	Integer32	dbStatHttpConnPoolMisses	HttpConnPoolMisses
1.3.6.1.4.1.897.2.2.1.72.n	Integer32	dbStatHttpConnPoolSteals	HttpConnPoolSteals
1.3.6.1.4.1.897.2.2.1.73.n	Integer32	dbStatMirrorServerWaits	MirrorServerWaits
1.3.6.1.4.1.897.2.2.1.74.n	Counter64	dbStatBytesReceived	BytesReceived
1.3.6.1.4.1.897.2.2.1.75.n	Counter64	dbStatBytesReceivedUn-comp	BytesReceivedUncomp
1.3.6.1.4.1.897.2.2.1.76.n	Counter64	dbStatBytesSent	BytesSent
1.3.6.1.4.1.897.2.2.1.77.n	Counter64	dbStatBytesSentUncomp	BytesSentUncomp
1.3.6.1.4.1.897.2.2.1.78.n	Integer32	dbStatCarverHeapPages	CarverHeapPages

OID	Type	Name	Statistic
1.3.6.1.4.1.897.2.2.1.79.n	Integer32	dbStatClientStmtCacheHits	ClientStmtCacheHits
1.3.6.1.4.1.897.2.2.1.80.n	Integer32	dbStatClientStmtCache-Misses	ClientStmtCacheMisses
1.3.6.1.4.1.897.2.2.1.81.n	Integer32	dbStatCommit	Commit
1.3.6.1.4.1.897.2.2.1.82.n	Integer32	dbStatCursor	Cursor
1.3.6.1.4.1.897.2.2.1.83.n	Integer32	dbStatHeapsCarver	HeapsCarver
1.3.6.1.4.1.897.2.2.1.84.n	Integer32	dbStatHeapsLocked	HeapsLocked
1.3.6.1.4.1.897.2.2.1.85.n	Integer32	dbStatHeapsQuery	HeapsQuery
1.3.6.1.4.1.897.2.2.1.86.n	Integer32	dbStatHeapsRelocatable	HeapsRelocatable
1.3.6.1.4.1.897.2.2.1.87.n	Integer32	dbStatCursorOpen	CursorOpen
1.3.6.1.4.1.897.2.2.1.88.n	Counter64	dbStatPacketsReceived	PacketsReceived
1.3.6.1.4.1.897.2.2.1.89.n	Counter64	dbStatPacketsReceivedUn-comp	PacketsReceivedUncomp
1.3.6.1.4.1.897.2.2.1.90.n	Counter64	dbStatPacketsSent	PacketsSent
1.3.6.1.4.1.897.2.2.1.91.n	Counter64	dbStatPacketsSentUncomp	PacketsSentUncomp
1.3.6.1.4.1.897.2.2.1.92.n	Integer32	dbStatQueryHeapPages	QueryHeapPages
1.3.6.1.4.1.897.2.2.1.93.n	Integer32	dbStatQueryMemActiveCurr	QueryMemActiveCurr
1.3.6.1.4.1.897.2.2.1.94.n	Integer32	dbStatQueryMemGrant-Failed	QueryMemGrantFailed
1.3.6.1.4.1.897.2.2.1.95.n	Integer32	dbStatQueryMemGrant-Granted	QueryMemGrantGranted
1.3.6.1.4.1.897.2.2.1.96.n	Integer32	dbStatQueryMemGrantRe-quested	QueryMemGrantRequested
1.3.6.1.4.1.897.2.2.1.97.n	Integer32	dbStatQueryMem-GrantWaited	QueryMemGrantWaited
1.3.6.1.4.1.897.2.2.1.98.n	Integer32	dbStatQueryMemGrantWait-ing	QueryMemGrantWaiting
1.3.6.1.4.1.897.2.2.1.99.n	Integer32	dbStatRequestsReceived	RequestsReceived
1.3.6.1.4.1.897.2.2.1.100.n	Integer32	dbStatRlbk	Rlbk
1.3.6.1.4.1.897.2.2.1.101.n	Integer32	dbStatPrepStmt	PrepStmt
1.3.6.1.4.1.897.2.2.1.102.n	Integer32	dbStatParameterizationPre-pareCount	ParameterizationPrepare-Count

Related Information

List of Database Server Properties [page 900]

1.8.2.3.8.6 SQL Anywhere MIB Database Properties

Retrieve database properties by using the SQL Anywhere SNMP Extension Agent.

Writable options are marked with an asterisk (*). The value `n` is the database number in the `sasnmp.ini` file.

Descriptions

OID	Type	Name	Property
1.3.6.1.4.1.897.2.2.2.1.n	String	dbPropAlias	Alias
1.3.6.1.4.1.897.2.2.2.2.n	String	dbPropAuditingTypes	AuditingTypes
1.3.6.1.4.1.897.2.2.2.3.n	String	dbPropBlankPadding	BlankPadding
1.3.6.1.4.1.897.2.2.2.4.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.5.n	String	dbPropCapabilities	Capabilities
1.3.6.1.4.1.897.2.2.2.6.n	String	dbPropCaseSensitive	CaseSensitive
1.3.6.1.4.1.897.2.2.2.7.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.8.n	String	dbPropCharSet	CharSet
1.3.6.1.4.1.897.2.2.2.9.n	String	dbPropChecksum	Checksum
1.3.6.1.4.1.897.2.2.2.10.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.11.n	String	dbPropCollation	Collation
1.3.6.1.4.1.897.2.2.2.12.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.13.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.14.n	String	dbPropCurrentRedoPos	CurrentRedoPos
1.3.6.1.4.1.897.2.2.2.15.n	String	dbPropDBFileFragments	DBFileFragments
1.3.6.1.4.1.897.2.2.2.16.n	String	dbPropDriveType	DriveType
1.3.6.1.4.1.897.2.2.2.17.n	String	dbPropEncryption	Encryption
1.3.6.1.4.1.897.2.2.2.18.n	String	dbPropFile	File
1.3.6.1.4.1.897.2.2.2.19.n	String	dbPropFileSize	FileSize
1.3.6.1.4.1.897.2.2.2.20.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.21.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.22.n	String	dbPropFreePages	FreePages
1.3.6.1.4.1.897.2.2.2.23.n	String	dbPropGlobalDBID	GlobalDBID
1.3.6.1.4.1.897.2.2.2.24.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.25.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.26.n	String	dbPropInternal	Internal

OID	Type	Name	Property
1.3.6.1.4.1.897.2.2.2.27.n	String	dbPropIQStore	IQStore
1.3.6.1.4.1.897.2.2.2.28.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.29.n	String	dbPropLanguage	Language
1.3.6.1.4.1.897.2.2.2.30.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.31.n	String	dbPropLogFileFragments	LogFileFragments
1.3.6.1.4.1.897.2.2.2.32.n	String	dbPropLogMirrorName	LogMirrorName
1.3.6.1.4.1.897.2.2.2.33.n	String	dbPropLogName	LogName
1.3.6.1.4.1.897.2.2.2.34.n	String	dbPropLTMGeneration	LTMGeneration
1.3.6.1.4.1.897.2.2.2.35.n	String	dbPropLTMTrunc	LTMTrunc
1.3.6.1.4.1.897.2.2.2.36.n	String	dbPropMultiByteCharSet	MultiByteCharSet
1.3.6.1.4.1.897.2.2.2.37.n	String	dbPropName	Name
1.3.6.1.4.1.897.2.2.2.38.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.39.n	String	dbPropPageSize	PageSize
1.3.6.1.4.1.897.2.2.2.40.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.41.n	String	dbPropProcedureProfiling*	ProcedureProfiling
1.3.6.1.4.1.897.2.2.2.42.n	String	dbPropReadOnly	ReadOnly
1.3.6.1.4.1.897.2.2.2.43.n	String	dbPropRemoteTrunc	RemoteTrunc
1.3.6.1.4.1.897.2.2.2.44.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.45.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.46.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.47.n	String	dbPropSyncTrunc	SyncTrunc
1.3.6.1.4.1.897.2.2.2.48.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.49.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.50.n	String	dbPropTempFileName	TempFileName
1.3.6.1.4.1.897.2.2.2.51.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.52.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.53.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.54.n	String	dbPropNextScheduleTime	NextScheduleTime
1.3.6.1.4.1.897.2.2.2.55.n	String	dbPropIdentitySignature	IdentitySignature
1.3.6.1.4.1.897.2.2.2.56.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.57.n	String	dbPropSnapshotIsolation-State	SnapshotIsolationState
1.3.6.1.4.1.897.2.2.2.58.n	String	dbPropConnsDisabled	ConnsDisabled
1.3.6.1.4.1.897.2.2.2.59.n	String	dbPropPartnerState	PartnerState

OID	Type	Name	Property
1.3.6.1.4.1.897.2.2.2.60.n	String	dbPropArbiterState	ArbiterState
1.3.6.1.4.1.897.2.2.2.61.n	String	dbPropMirrorState	MirrorState
1.3.6.1.4.1.897.2.2.2.62.n	String	dbPropAlternateServerName	AlternateServerName
1.3.6.1.4.1.897.2.2.2.63.n	String	dbPropEncryptionScope	EncryptionScope
1.3.6.1.4.1.897.2.2.2.64.n	String	dbPropNcharCharSet	NcharCharSet
1.3.6.1.4.1.897.2.2.2.65.n	String	dbPropNcharCollation	NcharCollation
1.3.6.1.4.1.897.2.2.2.66.n	String	dbPropAccentSensitive	AccentSensitive
1.3.6.1.4.1.897.2.2.2.67.n	String	dbPropSendingTracingTo	SendingTracingTo
1.3.6.1.4.1.897.2.2.2.68.n	String	dbPropReceivingTracingFrom	ReceivingTracingFrom
1.3.6.1.4.1.897.2.2.2.69.n	String	dbPropIOParallelism	IOParallelism
1.3.6.1.4.1.897.2.2.2.70.n	String	dbPropJavaVM	JavaVM
1.3.6.1.4.1.897.2.2.2.71.n	String	dbPropDatabaseCleaner	DatabaseCleaner
1.3.6.1.4.1.897.2.2.2.72.n	String	dbPropHasCollationTailoring	HasCollationTailoring
1.3.6.1.4.1.897.2.2.2.73.n	String	dbPropCatalogCollation	CatalogCollation
1.3.6.1.4.1.897.2.2.2.74.n	String	dbPropHasEndianSwapFix	HasEndianSwapFix
1.3.6.1.4.1.897.2.2.2.75.n	String	dbPropAlternateMirrorSer-verName	AlternateMirrorServerName
1.3.6.1.4.1.897.2.2.2.76.n	String	dbPropOptionWatchList	OptionWatchList
1.3.6.1.4.1.897.2.2.2.77.n	String	dbPropOptionWatchAction	OptionWatchAction
1.3.6.1.4.1.897.2.2.2.78.n	String	dbPropMirrorMode	MirrorMode
1.3.6.1.4.1.897.2.2.2.79.n	String	dbPropHasNCHARLegacy-CollationFix	HasNCHARLegacyCollation-Fix
1.3.6.1.4.1.897.2.2.2.80.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.81.n	String	dbPropAuthenticated	Authenticated
1.3.6.1.4.1.897.2.2.2.82.n	String	dbPropSynchronizationSche-maChangeActive	SynchronizationSchema-ChangeActive
1.3.6.1.4.1.897.2.2.2.83.n	String	dbPropLastCheckpointTime	LastCheckpointTime
1.3.6.1.4.1.897.2.2.2.84.n	String	dbPropMirrorServerState	MirrorServerState
1.3.6.1.4.1.897.2.2.2.85.n	String	dbPropDriveBus	DriveBus
1.3.6.1.4.1.897.2.2.2.86.n	String	dbPropDriveModel	DriveModel
1.3.6.1.4.1.897.2.2.2.87.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.88.n	String	dbPropMirrorRole	MirrorRole
1.3.6.1.4.1.897.2.2.2.89.n	String	dbPropWriteChecksum	WriteChecksum
1.3.6.1.4.1.897.2.2.2.90.n	String	dbPropApproximateCPU-Time	ApproximateCPUTime

OID	Type	Name	Property
1.3.6.1.4.1.897.2.2.2.91.n	String	dbPropConnectedTime	ConnectedTime
1.3.6.1.4.1.897.2.2.2.92.n	String	dbPropReqCountUnsched-uled	ReqCountUnscheduled
1.3.6.1.4.1.897.2.2.2.93.n	String	dbPropReqCountActive	ReqCountActive
1.3.6.1.4.1.897.2.2.2.94.n	String	dbPropReqCountBlockIO	ReqCountBlockIO
1.3.6.1.4.1.897.2.2.2.95.n	String	dbPropReqCountBlockLock	ReqCountBlockLock
1.3.6.1.4.1.897.2.2.2.96.n	String	dbPropReqCountBlockCon-tention	ReqCountBlockContention
1.3.6.1.4.1.897.2.2.2.97.n	String	dbPropReqTimeUnsched-uled	ReqTimeUnscheduled
1.3.6.1.4.1.897.2.2.2.98.n	String	dbPropReqTimeActive	ReqTimeActive
1.3.6.1.4.1.897.2.2.2.99.n	String	dbPropReqTimeBlockIO	ReqTimeBlockIO
1.3.6.1.4.1.897.2.2.2.100.n	String	dbPropReqTimeBlockLock	ReqTimeBlockLock
1.3.6.1.4.1.897.2.2.2.101.n	String	dbPropReqTimeBlockCon-tention	ReqTimeBlockContention
1.3.6.1.4.1.897.2.2.2.102.n	String	dbPropCopyNodeParent	CopyNodeParent
1.3.6.1.4.1.897.2.2.2.103.n	String	dbPropLastCommitRedoPos	LastCommitRedoPos
1.3.6.1.4.1.897.2.2.2.104.n	String	dbPropLastWrittenRedoPos	LastWrittenRedoPos
1.3.6.1.4.1.897.2.2.2.105.n	String	dbPropLastSyncedRedoPos	LastSyncedRedoPos
1.3.6.1.4.1.897.2.2.2.106.n	String	dbPropUTCTimestampCata-log	UTCTimestampCatalog
1.3.6.1.4.1.897.2.2.2.107.n	String	dbPropDiskSandbox	DiskSandbox
1.3.6.1.4.1.897.2.2.2.108.n	String	dbPropTimeWithoutClient-Connection	TimeWithoutClientConnec-tion
1.3.6.1.4.1.897.2.2.2.109.n	String	dbPropBackupInProgress	BackupInProgress
1.3.6.1.4.1.897.2.2.2.110.n	String	dbPropIQPointInTimeReco-veryLogStatus	IQPointInTimeRecoveryLog-Status
1.3.6.1.4.1.897.2.2.2.111.n	String	dbPropPointInTimeRecovery	PointInTimeRecovery
1.3.6.1.4.1.897.2.2.2.112.n	String	dbPropHasQDADTTFeature	HasQDADTTFeature
1.3.6.1.4.1.897.2.2.2.113.n	String	dbPropIQPointInTimeReco-veryLogFilename	IQPointInTimeRecoveryLog-Filename
1.3.6.1.4.1.897.2.2.2.114.n	String	dbPropMaxConnections	MaxConnections
1.3.6.1.4.1.897.2.2.2.115.n	String	dbPropTimeZone	TimeZone
1.3.6.1.4.1.897.2.2.2.116.n	String	dbPropCurrentTimeZoneOff-set	CurrentTimeZoneOffset
1.3.6.1.4.1.897.2.2.2.117.n	String	dbPropCurrentTimelineID	CurrentTimelineID
1.3.6.1.4.1.897.2.2.2.118.n	String	dbPropInternal	Internal

OID	Type	Name	Property
1.3.6.1.4.1.897.2.2.2.119.n	String	dbPropPreviousTimelineID	PreviousTimelineID
1.3.6.1.4.1.897.2.2.2.120.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.121.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.122.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.123.n	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.124.n	String	dbPropIdentitySignatur-eUUID	IdentitySignatureUUID
1.3.6.1.4.1.897.2.2.2.125.n	String	dbPropPlanStatisticsStored	PlanStatisticsStored
1.3.6.1.4.1.897.2.2.2.126.n	String	dbPropCurrentTimeline-Signature	CurrentTimelineSignature
1.3.6.1.4.1.897.2.2.2.127.n	String	dbPropPropertyHistoryList	PropertyHistoryList

Related Information

List of Database Server Properties [page 900]

1.8.2.3.8.7 SQL Anywhere MIB Database Options

Retrieve database options by using the SQL Anywhere SNMP Extension Agent.

Writable options are marked with an asterisk (*). The value `n` is the database number in the `sasnmp.ini` file.

Descriptions

OID	Type	Name	Option
1.3.6.1.4.1.897.2.2.3.1.n	String	dbOptAllowNullsByDefault*	allow_nulls_by_default
1.3.6.1.4.1.897.2.2.3.2.n	String	dbOptAnsinull*	ansinull
1.3.6.1.4.1.897.2.2.3.3.n	String	dbOptAnsiBlanks*	ansi_blanks
1.3.6.1.4.1.897.2.2.3.4.n	String	dbOptAnsiCloseCursorsOn-Rollback*	ansi_close_cursors_on_roll-back
1.3.6.1.4.1.897.2.2.3.5.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.6.n	String	dbOptAnsiPermissions*	ansi_permissions
1.3.6.1.4.1.897.2.2.3.7.n	String	dbOptAnsiUpdateCon-straints*	ansi_update_constraints

OID	Type	Name	Option
1.3.6.1.4.1.897.2.2.3.8.n	String	dbOptAuditing*	auditing
1.3.6.1.4.1.897.2.2.3.9.n	String	dbOptAuditingOptions*	auditing_options
1.3.6.1.4.1.897.2.2.3.10.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.11.n	String	dbOptBackgroundPriority*	background_priority
1.3.6.1.4.1.897.2.2.3.12.n	String	dbOptBlocking*	blocking
1.3.6.1.4.1.897.2.2.3.13.n	Integer32	dbOptBlockingTimeout*	blocking_timeout
1.3.6.1.4.1.897.2.2.3.14.n	String	dbOptChained*	chained
1.3.6.1.4.1.897.2.2.3.15.n	Integer32	dbOptCheckpointTime*	checkpoint_time
1.3.6.1.4.1.897.2.2.3.16.n	Integer32	dbOptCisOption*	cis_option
1.3.6.1.4.1.897.2.2.3.17.n	Integer32	dbOptCisRowsetSize*	cis_rowset_size
1.3.6.1.4.1.897.2.2.3.18.n	String	dbOptCloseOnEndtrans*	close_on_endtrans
1.3.6.1.4.1.897.2.2.3.19.n	String	dbOptConnectionAuthentication*	connection_authentication
1.3.6.1.4.1.897.2.2.3.20.n	String	dbOptContinueAfterRaiserror*	continue_after_raiserror
1.3.6.1.4.1.897.2.2.3.21.n	String	dbOptConversionError*	conversion_error
1.3.6.1.4.1.897.2.2.3.22.n	String	dbOptCooperativeCommits*	cooperative_commits
1.3.6.1.4.1.897.2.2.3.23.n	Integer32	dbOptCooperativeCommitTimeout*	cooperative_commit_timeout
1.3.6.1.4.1.897.2.2.3.24.n	String	dbOptDatabaseAuthentication*	database_authentication
1.3.6.1.4.1.897.2.2.3.25.n	String	dbOptDateFormat*	date_format
1.3.6.1.4.1.897.2.2.3.26.n	String	dbOptDateOrder*	date_order
1.3.6.1.4.1.897.2.2.3.27.n	String	dbOptDebugMessages*	debug_messages
1.3.6.1.4.1.897.2.2.3.28.n	String	dbOptDedicatedTask*	dedicated_task
1.3.6.1.4.1.897.2.2.3.29.n	Integer32	dbOptDefaultTimestampIncrement*	default_timestamp_increment
1.3.6.1.4.1.897.2.2.3.30.n	String	dbOptDelayedCommits*	delayed_commits
1.3.6.1.4.1.897.2.2.3.31.n	Integer32	dbOptDelayedCommitTimeout*	delayed_commit_timeout
1.3.6.1.4.1.897.2.2.3.32.n	String	dbOptDivideByZeroError*	divide_by_zero_error
1.3.6.1.4.1.897.2.2.3.33.n	String	dbOptEscapeCharacter*	escape_character
1.3.6.1.4.1.897.2.2.3.34.n	String	dbOptExcludeOperators*	exclude_operators
1.3.6.1.4.1.897.2.2.3.35.n	String	dbOptExtendedJoinSyntax*	extended_join_syntax
1.3.6.1.4.1.897.2.2.3.36.n	String	dbOptFireTriggers*	fire_triggers
1.3.6.1.4.1.897.2.2.3.37.n	Integer32	dbOptFirstDayOfWeek*	first_day_of_week

OID	Type	Name	Option
1.3.6.1.4.1.897.2.2.3.38.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.39.n	String	dbOptForceViewCreation*	force_view_creation
1.3.6.1.4.1.897.2.2.3.40.n	String	dbOptForXmlNullTreatment*	for_xml_null_treatment
1.3.6.1.4.1.897.2.2.3.41.n	Integer32	dbOptGlobalDatabaseId*	global_database_id
1.3.6.1.4.1.897.2.2.3.42.n	Integer32	dbOptIsolationLevel*	isolation_level
1.3.6.1.4.1.897.2.2.3.43.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.44.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.45.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.46.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.47.n	String	dbOptLockRejectedRows*	lock_rejected_rows
1.3.6.1.4.1.897.2.2.3.48.n	String	dbOptLoginMode*	login_mode
1.3.6.1.4.1.897.2.2.3.49.n	String	dbOptLoginProcedure*	login_procedure
1.3.6.1.4.1.897.2.2.3.50.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.51.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.52.n	Integer32	dbOptMaxCursorCount*	max_cursor_count
1.3.6.1.4.1.897.2.2.3.53.n	Integer32	dbOptMaxHashSize*	max_hash_size
1.3.6.1.4.1.897.2.2.3.54.n	Integer32	dbOptMaxPlansCached*	max_plans_cached
1.3.6.1.4.1.897.2.2.3.55.n	Integer32	dbOptMaxRecursiveIterations*	max_recursive_iterations
1.3.6.1.4.1.897.2.2.3.56.n	Integer32	dbOptMaxStatementCount*	max_statement_count
1.3.6.1.4.1.897.2.2.3.57.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.58.n	Integer32	dbOptMinPasswordLength*	min_password_length
1.3.6.1.4.1.897.2.2.3.59.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.60.n	Integer32	dbOptNearestCentury*	nearest_century
1.3.6.1.4.1.897.2.2.3.61.n	String	dbOptNonKeywords*	non_keywords
1.3.6.1.4.1.897.2.2.3.62.n	String	dbOptOdbcDistinguishCharAndVarchar*	odbc_distinguish_char_and_varchar
1.3.6.1.4.1.897.2.2.3.63.n	String	dbOptOnCharsetConversionFailure*	on_charset_conversion_failure
1.3.6.1.4.1.897.2.2.3.64.n	String	dbOptOnTsqlError*	on_tsql_error
1.3.6.1.4.1.897.2.2.3.65.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.66.n	String	dbOptOptimizationGoal*	optimization_goal
1.3.6.1.4.1.897.2.2.3.67.n	Integer32	dbOptOptimizationLevel*	optimization_level
1.3.6.1.4.1.897.2.2.3.68.n	Integer32	dbOptInternal	Internal

OID	Type	Name	Option
1.3.6.1.4.1.897.2.2.3.69.n	String	dbOptOptimizationWorkload*	optimization_workload
1.3.6.1.4.1.897.2.2.3.70.n	Integer32	dbOptPinnedCursorPercentOfCache*	pinned_cursor_percent_of_cache
1.3.6.1.4.1.897.2.2.3.71.n	Integer32	dbOptPrecision*	precision
1.3.6.1.4.1.897.2.2.3.72.n	String	dbOptPrefetch*	prefetch
1.3.6.1.4.1.897.2.2.3.73.n	String	dbOptPreserveSourceFormat*	preserve_source_format
1.3.6.1.4.1.897.2.2.3.74.n	String	dbOptPreventArticlePkeyUpdate*	prevent_article_pkey_update
1.3.6.1.4.1.897.2.2.3.75.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.76.n	String	dbOptQuotedIdentifier*	quoted_identifier
1.3.6.1.4.1.897.2.2.3.77.n	String	dbOptReadPastDeleted*	read_past_deleted
1.3.6.1.4.1.897.2.2.3.78.n	Integer32	dbOptRecoveryTime*	recovery_time
1.3.6.1.4.1.897.2.2.3.79.n	String	dbOptReplicateAll*	replicate_all
1.3.6.1.4.1.897.2.2.3.80.n	String	dbOptReturnDateTimeAsString*	return_date_time_as_string
1.3.6.1.4.1.897.2.2.3.81.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.82.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.83.n	String	dbOptRowCounts*	row_counts
1.3.6.1.4.1.897.2.2.3.84.n	Integer32	dbOptScale*	scale
1.3.6.1.4.1.897.2.2.3.85.n	String	dbOptSortCollation*	sort_collation
1.3.6.1.4.1.897.2.2.3.86.n	String	dbOptSqlFlaggerErrorLevel*	sql_flagger_error_level
1.3.6.1.4.1.897.2.2.3.87.n	String	dbOptSqlFlaggerWarningLevel*	sql_flagger_warning_level
1.3.6.1.4.1.897.2.2.3.88.n	String	dbOptStringRtruncation*	string_rtruncation
1.3.6.1.4.1.897.2.2.3.89.n	String	dbOptSubsumeRowLocks*	subsume_row_locks
1.3.6.1.4.1.897.2.2.3.90.n	String	dbOptSuppressTdsDebugging*	suppress_tds_debugging
1.3.6.1.4.1.897.2.2.3.91.n	String	dbOptTdsEmptyStringIsNull*	tds_empty_string_is_null
1.3.6.1.4.1.897.2.2.3.92.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.93.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.94.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.95.n	String	dbOptTimestampFormat*	timestamp_format
1.3.6.1.4.1.897.2.2.3.96.n	String	dbOptTimeFormat*	time_format
1.3.6.1.4.1.897.2.2.3.97.n	Integer32	dbOptTimeZoneAdjustment*	time_zone_adjustment

OID	Type	Name	Option
1.3.6.1.4.1.897.2.2.3.98.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.99.n	String	dbOptTruncateTimestamp-Values*	truncate_timestamp_values
1.3.6.1.4.1.897.2.2.3.100.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.101.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.102.n	String	dbOptTsqlVariables*	tsql_variables
1.3.6.1.4.1.897.2.2.3.103.n	String	dbOptUpdateStatistics*	update_statistics
1.3.6.1.4.1.897.2.2.3.104.n	String	dbOptUpgradeDatabaseCap-ability*	upgrade_database_capability
1.3.6.1.4.1.897.2.2.3.105.n	String	dbOptUserEstimates*	user_estimates
1.3.6.1.4.1.897.2.2.3.106.n	String	dbOptWaitForCommit*	wait_for_commit
1.3.6.1.4.1.897.2.2.3.107.n	String	dbOptTempSpaceLimitCh-eck*	temp_space_limit_check
1.3.6.1.4.1.897.2.2.3.108.n	Integer32	dbOptRemoteIdleTimeout*	remote_idle_timeout
1.3.6.1.4.1.897.2.2.3.109.n	String	dbOptAnsiSubstring*	ansi_substring
1.3.6.1.4.1.897.2.2.3.110.n	String	dbOptOdbcDescribeBinar-yAsVarbinary*	odbc_describe_bi-nary_as_varbinary
1.3.6.1.4.1.897.2.2.3.111.n	String	dbOptRollbackOnDeadlock*	rollback_on_deadlock
1.3.6.1.4.1.897.2.2.3.112.n	String	dbOptIntegratedServer-Name*	integrated_server_name
1.3.6.1.4.1.897.2.2.3.113.n	String	dbOptLogDeadlocks*	log_deadlocks
1.3.6.1.4.1.897.2.2.3.114.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.115.n	String	dbOptWebserviceNamespa-ceHost*	webservice_namespace_host
1.3.6.1.4.1.897.2.2.3.116.n	Integer32	dbOptMaxQueryTasks*	max_query_tasks
1.3.6.1.4.1.897.2.2.3.117.n	Integer32	dbOptRequestTimeout*	request_timeout
1.3.6.1.4.1.897.2.2.3.118.n	String	dbOptSynchronizeMirrorOn-Commit*	synchronize_mirror_on_com-mit
1.3.6.1.4.1.897.2.2.3.119.n	Integer32	dbOptHttpSessionTimeout*	http_session_timeout
1.3.6.1.4.1.897.2.2.3.120.n	String	dbOptUuidHasHyphens*	uuid_has_hyphens
1.3.6.1.4.1.897.2.2.3.121.n	String	dbOptAllowSnapshotIsola-tion*	allow_snapshot_isolation
1.3.6.1.4.1.897.2.2.3.122.n	String	dbOptVerifyPasswordFunc-tion*	verify_password_function
1.3.6.1.4.1.897.2.2.3.123.n	String	dbOptDefaultDbspace*	default_dbspace
1.3.6.1.4.1.897.2.2.3.124.n	String	dbOptCollectStatistic-sOnDmlUpdates*	collect_statis-tics_on_dml_updates
1.3.6.1.4.1.897.2.2.3.125.n	String	dbOptInternal	Internal

OID	Type	Name	Option
1.3.6.1.4.1.897.2.2.3.126.n	String	dbOptJavaLocation*	java_location
1.3.6.1.4.1.897.2.2.3.127.n	String	dbOptOemString*	oem_string
1.3.6.1.4.1.897.2.2.3.128.n	Integer32	dbOptMaxTempSpace*	max_temp_space
1.3.6.1.4.1.897.2.2.3.129.n	String	dbOptSecureFeatureKey*	secure_feature_key
1.3.6.1.4.1.897.2.2.3.130.n	String	dbOptMaterializedViewOptimization*	materialized_view_optimization
1.3.6.1.4.1.897.2.2.3.131.n	Integer32	dbOptUpdatableStatementIsolation*	updatable_statement_isolation
1.3.6.1.4.1.897.2.2.3.132.n	String	dbOptTsqlOuterJoins*	tsql_outer_joins
1.3.6.1.4.1.897.2.2.3.133.n	String	dbOptPostLoginProcedure*	post_login_procedure
1.3.6.1.4.1.897.2.2.3.134.n	String	dbOptConnAuditing*	conn_auditing
1.3.6.1.4.1.897.2.2.3.135.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.136.n	String	dbOptJavaVmOptions*	java_vm_options
1.3.6.1.4.1.897.2.2.3.137.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.138.n	Integer32	dbOptMaxClientStatementsCached*	max_client_statements_cached
1.3.6.1.4.1.897.2.2.3.139.n	String	dbOptQueryMemTimeout*	query_mem_timeout
1.3.6.1.4.1.897.2.2.3.140.n	String	dbOptAllowReadClientFile*	allow_read_client_file
1.3.6.1.4.1.897.2.2.3.141.n	String	dbOptAllowWriteClientFile*	allow_write_client_file
1.3.6.1.4.1.897.2.2.3.142.n	String	dbOptPriority*	priority
1.3.6.1.4.1.897.2.2.3.143.n	String	dbOptMaxPriority*	max_priority
1.3.6.1.4.1.897.2.2.3.144.n	String	dbOptProgressMessages*	progress_messages
1.3.6.1.4.1.897.2.2.3.145.n	Integer32	dbOptBlockingOthersTimeout*	blocking_others_timeout
1.3.6.1.4.1.897.2.2.3.146.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.147.n	String	dbOptTimestampWithTimeZoneFormat*	timestamp_with_time_zone_format
1.3.6.1.4.1.897.2.2.3.148.n	Integer32	dbOptHttpConnectionPoolTimeout*	http_connection_pool_timeout
1.3.6.1.4.1.897.2.2.3.149.n	Integer32	dbOptHttpConnectionPoolBasesize*	http_connection_pool_basesize
1.3.6.1.4.1.897.2.2.3.150.n	String	dbOptStGeometryDescribeType*	st_geometry_describe_type
1.3.6.1.4.1.897.2.2.3.151.n	String	dbOptStGeometryAstextFormat*	st_geometry_astext_format
1.3.6.1.4.1.897.2.2.3.152.n	String	dbOptStGeometryAsbinaryFormat*	st_geometry_asbinary_format

OID	Type	Name	Option
1.3.6.1.4.1.897.2.2.3.153.n	String	dbOptStGeometryAsxmlFormat*	st_geometry_asxml_format
1.3.6.1.4.1.897.2.2.3.154.n	String	dbOptReservedKeywords*	reserved_keywords
1.3.6.1.4.1.897.2.2.3.155.n	String	dbOptStGeometryOnInvalid*	st_geometry_on_invalid
1.3.6.1.4.1.897.2.2.3.156.n	Integer32	dbOptMinRoleAdmins*	min_role_admins
1.3.6.1.4.1.897.2.2.3.157.n	String	dbOptTrustedCertificatesFile*	trusted_certificates_file
1.3.6.1.4.1.897.2.2.3.158.n	String	dbOptJavaClassPath*	java_class_path
1.3.6.1.4.1.897.2.2.3.159.n	String	dbOptStGeometryInterpolation*	st_geometry_interpolation
1.3.6.1.4.1.897.2.2.3.160.n	String	dbOptExternLoginCredentials*	extern_login_credentials
1.3.6.1.4.1.897.2.2.3.161.n	String	dbOptWebserviceSessionidName*	webservice_sessionid_name
1.3.6.1.4.1.897.2.2.3.162.n	String	dbOptDbPublisher*	db_publisher
1.3.6.1.4.1.897.2.2.3.163.n	String	dbOptAutoCommitOnCreateLocalTempIndex*	auto_commit_on_create_local_temp_index
1.3.6.1.4.1.897.2.2.3.164.n	String	dbOptDiskSandbox*	disk_sandbox
1.3.6.1.4.1.897.2.2.3.165.n	String	dbOptBaseTablesInRlvStore*	base_tables_in_rlv_store
1.3.6.1.4.1.897.2.2.3.166.n	String	dbOptAuditLog*	audit_log
1.3.6.1.4.1.897.2.2.3.167.n	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.168.n	Integer32	dbOptMaxConnections*	max_connections
1.3.6.1.4.1.897.2.2.3.169.n	String	dbOptTimeZone*	time_zone
1.3.6.1.4.1.897.2.2.3.170.n	String	dbOptConnectionType*	connection_type
1.3.6.1.4.1.897.2.2.3.171.n	String	dbOptParameterizationLevel*	parameterization_level
1.3.6.1.4.1.897.2.2.3.172.n	String	dbOptAutoCommit*	auto_commit

Related Information

Alphabetical List of Database Options [page 663]

1.8.2.3.9 RDBMS MIB Reference

The OIDs of many values can be retrieved using the SQL Anywhere SNMP Extension Agent.

By default, the RDBMS MIB is located in *C:\Program Files\SQL Anywhere 17*`\snmp\RDBMS-MIB.mib`.

In this section:

1.8.2.3.9.1 rdbmsDbTable

Lists information about the databases installed on a system.

The value db is the database number in the `sasnmp.ini` file.

OID	Type	Name	Value returned
1.3.6.1.2.1.39.1.1.1.1.db	Integer	rdbmsDbIndex	db
1.3.6.1.2.1.39.1.1.1.2.db	OID	rdbmsDbPrivateMibOID	1.3.6.1.4.1.897.2
1.3.6.1.2.1.39.1.1.1.3.db	String	rdbmsDbVendorName	PROPERTY('CompanyName')
1.3.6.1.2.1.39.1.1.1.4.db	String	rdbmsDbName	DB_PROPERTY('Name')
1.3.6.1.2.1.39.1.1.1.5.db	String	rdbmsDbContact	PROPERTY('LicensedUser')

1.8.2.3.9.2 rdbmsDbInfoTable

Provides additional information about the databases on the system.

The value db is the database number in the `sasnmp.ini` file.

OID	Type	Name	Value returned
1.3.6.1.2.1.39.1.2.1.1.db	String	rdbmsDbInfoProductName	PROPERTY('ProductName')
1.3.6.1.2.1.39.1.2.1.2.db	String	rdbmsDbInfoVersion	PROPERTY('ProductVersion')
1.3.6.1.2.1.39.1.2.1.3.db	Integer	rdbmsDbInfoSizeUnits	Calculated based on dbInfoSizeAllocated and dbInfoSizeUsed. • 1=bytes • 2=KB • 3=MB • 4=GB • 5=TB (Each unit is 1024 times the previous.)
1.3.6.1.2.1.39.1.2.1.4.db	Integer	rdbmsDbInfoSizeAllocated	DB_PROPERTY('PageSize') * DB_PROPERTY('FileSize')
1.3.6.1.2.1.39.1.2.1.5.db	Integer	rdbmsDbInfoSizeUsed	DB_PROPERTY('PageSize') * (DB_PROPERTY('FileSize') - DB_PROPERTY('FreePages'))
1.3.6.1.2.1.39.1.2.1.6.db	String	rdbmsDbInfoLastBackup	NULL[1]

[1] This OID is not supported by the SQL Anywhere SNMP Extension Agent.

1.8.2.3.9.3 rdbmsDbParamTable

Lists the configuration parameters for the databases on the system.

The value db is the database number in the `sasnmp.ini` file, while n is the index of the option in the `sa.2.3` subtree.

OID	Type	Name	Value returned
1.3.6.1.2.1.39.1.3.1.1.db	String	rdbmsDbParamName	Option name
1.3.6.1.2.1.39.1.3.1.2.db	Integer	rdbmsDbParamSubIndex	n
1.3.6.1.2.1.39.1.3.1.3.db	OID	rdbmsDbParamID	OID in SQL Anywhere MIB corresponding to this option
1.3.6.1.2.1.39.1.3.1.4.db	String	rdbmsDbParamCurrValue	Option value
1.3.6.1.2.1.39.1.3.1.5.db	String	rdbmsDbParamComment	NULL[1]

[1] This OID is not supported by the SQL Anywhere SNMP Extension Agent.

1.8.2.3.9.4 rdbmsDbLimitedResourceTable

Lists free space information about each dbspace.

In this table, n represents each dbspace as follows:

- 1-13 are for normal dbspaces (numbered 0-12 in the database)
- 14 is the transaction log file
- 15 is the transaction log mirror file
- 16 is the temporary file

The value db is the database number in the `sasnmp.ini` file.

OID	Type	Name	Value returned
1.3.6.1.2.1.39.1.4.1.1.n.db	String	rdbmsDbLimitedResource-Name	Name of dbspace, or Transaction Log, Transaction Log Mirror, or Temporary File.
1.3.6.1.2.1.39.1.4.1.2.n.db	OID	rdbmsDbLimitedResourceID	1.3.6.1.4.1.897.2
1.3.6.1.2.1.39.1.4.1.3.n.db	Integer	rdbmsDbLimitedResourceLimit	Free space available on disk + current file size
1.3.6.1.2.1.39.1.4.1.4.n.db	Integer	rdbmsDbLimitedResource-Current	Current file size
1.3.6.1.2.1.39.1.4.1.5.n.db	Integer	rdbmsDbLimitedResource-Highwater	Current size
1.3.6.1.2.1.39.1.4.1.6.n.db	Integer	rdbmsDbLimitedResource-Failure	0[1]
1.3.6.1.2.1.39.1.4.1.7.n.db	String	rdbmsDbLimitedResource-Description	One of Bytes, KB, MB, GB, or TB.

[1] This OID is not supported by the SQL Anywhere SNMP Extension Agent.

1.8.2.3.9.5 rdbmsSrvTable

Lists the database servers running or installed on your system.

The value db is the database number in the `sasnmp.ini` file.

OID	Type	Name	Value returned
1.3.6.1.2.1.39.1.5.1.1.db	OID	rdbmsSrvPrivateMibOID	1.3.6.1.4.1.897.2
1.3.6.1.2.1.39.1.5.1.2.db	String	rdbmsSrvVendorName	PROPERTY('CompanyName')
1.3.6.1.2.1.39.1.5.1.3.db	String	rdbmsSrvProductName	PROPERTY('ProductName')
1.3.6.1.2.1.39.1.5.1.4.db	String	rdbmsSrvContact	PROPERTY('LicensedCompany')

1.8.2.3.9.6 rdbmsSrvInfoTable

Lists additional information about the database servers in your system.

The value db is the database number in the sasnmp.ini file.

OID	Type	Name	Value returned
1.3.6.1.2.1.39.1.6.1.1.db	Integer	rdbmsSrvInfoStartupTime	PROPERTY('StartTime')
1.3.6.1.2.1.39.1.6.1.2.db	Integer	rdbmsSrvInfoFinishedTransactions	0[1]
1.3.6.1.2.1.39.1.6.1.3.db	Integer	rdbmsSrvInfoDiskReads	PROPERTY('DiskReadEng')
1.3.6.1.2.1.39.1.6.1.4.db	Integer	rdbmsSrvInfoLogicalReads	0[1]
1.3.6.1.2.1.39.1.6.1.5.db	Integer	rdbmsSrvInfoDiskWrites	PROPERTY('DiskWriteEng')
1.3.6.1.2.1.39.1.6.1.6.db	Integer	rdbmsSrvInfoLogicalWrites	0[1]
1.3.6.1.2.1.39.1.6.1.7.db	Integer	rdbmsSrvInfoPageReads	0[1]
1.3.6.1.2.1.39.1.6.1.8.db	Integer	rdbmsSrvInfoPageDiskOutOfWrites	0[1]
1.3.6.1.2.1.39.1.6.1.9.db	Integer	rdbmsSrvInfoSpaces	0[1]
1.3.6.1.2.1.39.1.6.1.10.db	Integer	rdbmsSrvInfoHandledRequests	PROPERTY('Req')
1.3.6.1.2.1.39.1.6.1.11.db	Integer	rdbmsSrvInfoRequestRecvs	PROPERTY('PacketsReceivedUncomp')
1.3.6.1.2.1.39.1.6.1.12.db	Integer	rdbmsSrvInfoRequestSends	PROPERTY('PacketsSentUncomp')
1.3.6.1.2.1.39.1.6.1.13.db	Integer	rdbmsSrvInfoHighwaterInboundAssociations	0[1]
1.3.6.1.2.1.39.1.6.1.14.db	Integer	rdbmsSrvInfoMaxInboundAssociations	0[1]

[1] This OID is not supported by the SQL Anywhere SNMP Extension Agent.

1.8.2.3.9.7 rdbmsSrvParamTable

Lists the server options that can be set by the SQL Anywhere SNMP Extension Agent through the SQL Anywhere MIB.

n is the index, as follows:

n	Server option
1	ConnsDisabled
2	LivenessTimeout (default)
3	QuittingTime

n	Server option
4	RememberLastStatement
5	RequestLogFile
6	RequestLogging

The value db is the database number in the sasnmp.ini file.

OID	Type	Name	Value returned
1.3.6.1.2.1.39.1.7.1.1.n.db	String	rdbmsDbSrvParamName	Name of option n
1.3.6.1.2.1.39.1.7.1.2.n.db	Integer	rdbmsDbSrvParamSubIndex	n
1.3.6.1.2.1.39.1.7.1.3.n.db	OID	rdbmsDbSrvParamID	1.3.6.1.4.1.897.2
1.3.6.1.2.1.39.1.7.1.4.n.db	String	rdbmsDbSrvParamCurrValue	Current value of option n
1.3.6.1.2.1.39.1.7.1.5.n.db	String	rdbmsDbSrvParamComment	Full name of option n

1.8.2.3.9.8 rdbmsSrvLimitedResourceTable

Contains information about server configuration parameters.

The value db is the database number in the sasnmp.ini file, while n is the index of the resource as follows:

n	Name	Resource	Resource limit
1	Connections	PROPERTY('UniqueClientAddresses')	PROPERTY('LicenseCount')
2	Processors	PROPERTY('NumLogicalProcessorsUsed')	PROPERTY('NumLogicalProcessorsUsed')

OID	Type	Name	Value returned
1.3.6.1.2.1.39.1.8.1.1.db	String	rdbmsSrvLimitedResourceName	Name of resource n
1.3.6.1.2.1.39.1.8.1.2.db	OID	rdbmsSrvLimitedResourceID	OID in SQL Anywhere MIB corresponding to this option
1.3.6.1.2.1.39.1.8.1.3.db	Integer	rdbmsSrvLimitedResourceLimit	Upper limit of resource n
1.3.6.1.2.1.39.1.8.1.4.db	Integer	rdbmsSrvLimitedResourceCurrent	Current value of resource n
1.3.6.1.2.1.39.1.8.1.5.db	Integer	rdbmsSrvLimitedResourceHighwater	Current value of resource n
1.3.6.1.2.1.39.1.8.1.6.db	Integer	rdbmsSrvLimitedResourceFailures	0[1]
1.3.6.1.2.1.39.1.8.1.7.db	String	rdbmsSrvLimitedResourceDescription	Name of resource n

[1] This OID is not supported by the SQL Anywhere SNMP Extension Agent.

1.8.3 Performance

You can use a number of tools to improve performance of a database and database server.

In this section:

1.8.3.1 SQL Anywhere Cockpit

The Cockpit is a database server monitoring tool that provides an up-to-date view of the availability, capacity, and performance of your database server. Use a browser to access it. It also provides information for the SAP DB Control Center.

The Cockpit issues alerts when it detects conditions that could indicate possible problems in your system. There are predefined alerts for conditions such as high CPU use, a high number of database server connections, and lengthy blocked connections and operations. From the Cockpit, you control the threshold values for the alerts, and email notification settings. Alerts exist while the conditions that create the alert are true; when these conditions are no longer present, the alert disappears.

Use the Cockpit to view the users who are connected to a database server, and to display both database server and database properties. You can perform simple administration tasks such as starting backups and dropping connections.

The Cockpit runs on the same database server that it monitors. It is designed to run continuously in the background so that you can connect to it when needed. But, you can also easily start and stop it to investigate issues.

Start the Cockpit when you start your database server, so that it is always available when you need to view the current state of the database server.

Connect to the Cockpit by using the credentials of a user from one of the databases running on the database server. This user must have the COCKPIT_ROLE user-defined role. In newly created databases, only the administrator rights are granted to the database administrator, so you must grant this user-defined role to any user that needs to connect to the Cockpit. To perform tasks using the Cockpit, you need the associated roles or privileges. For example, to back up a database, you must also have the BACKUP DATABASE system privilege for that database.

For information about the browsers that the Cockpit supports, see SAP SQL Anywhere Components by Platform.

When you start the Cockpit, you must decide whether you want to preserve your settings and alert histories between sessions.

The Cockpit application uses an embedded database, referred to as the Cockpit database. This database stores the configuration settings, such as alert thresholds, and alert histories. The Cockpit database can be a permanent database, for which you specify a file name and path, or a temporary database. A temporary Cockpit database is deleted when the Cockpit stops. When the Cockpit database is a permanent database any changes you make within the Cockpit are automatically saved to the file.

When you start the Cockpit, you choose whether to use a permanent database or a temporary database. If you choose to use a temporary database, you can switch to using a permanent database from the Cockpit by opening *MONITORING* page, clicking *Configure Alerts*, and then clicking *Switch to a permanent Cockpit database*.

The CockpitDB database server property lists the set of Cockpit options currently being used by the database server, as well as the Temp parameter. When Temp is set to yes, the Cockpit database is a temporary database.

You can view the CockpitDB database server property by opening the *HOME* page, clicking *Show Details* and locating the property in the properties table.

In this section:

Related Information

Upgrading Version 16 Databases to Use the SQL Anywhere Cockpit
-cdb Database Server Option [page 405]
sa_server_option System Procedure
COCKPIT_ROLE User-defined Role [page 1544]

1.8.3.1.1 Starting and Connecting to the Cockpit(SQL Central)

Start the Cockpit on a running database server and use it to view the current state of your database server.

Prerequisites

To stop and start the Cockpit, you need the EXECUTE system privilege for the sa_server_option system procedure and the SERVER OPERATOR system privilege.

The Cockpit runs on the database server that it monitors. You connect to the Cockpit using the credentials (database user ID and password) from a database that is running on the database server.

The database server must:

- Accept shared memory connections. For example, if the command to start the database server includes the -ec TLS database server option, then include the -es database server option.
- Have at least one version 16 or later database running on it.
- Have an HTTPS connection listener, so that the communication between the Cockpit and the web browser is encrypted. Otherwise, you risk a third party observing the communications between the Cockpit and web browser, and obtaining sensitive information including the database user name and password that you use to connect to the Cockpit.
  An HTTPS connection listener requires a certificate that must be either signed using a certificate that you create or signed by a certificate authority. Certificate authorities generally charge a fee to sign a certificate. You can create a signed certificate using the Create Certificate Creation utility (createcert) by creating a root certificate and then using it to create the signed certificate. Update your browser's list of trusted root certificates to include your root certificate file. This step is not required when you purchase the signed certificate from a certificate authority because browsers maintain a list of trusted certificates from common certificate authorities.
  Start an HTTPS connection listener when you start the database by specifying the -xs database server option and by specifying HTTPS as the protocol to start an HTTPS connection listener. For example:

```
dbsrv17 -xs "HTTPS(PORT=443;IDENTITY=c:\my-signed-certificate-identity-
file.id;IDENTITY_PASSWORD=my-signed-certificate-password)" "C:\Users\Public
\Documents\SQL Anywhere
    17\Samples\demo.db"
```

  By default, an HTTPS connection listener is shared with all databases running on the database server. To restrict an HTTPS connection listener to only the Cockpit, use the DBN protocol option and specify *SQLACockpit* as the database-name. For example:

```
dbsrv17 -cdb "DBF=C:\temp\cockpitsettings.db" -xs
"HTTPS(PORT=443;DBN=SQLACockpit;IDENTITY=c:\my-signed-certificate-identity-
file.id;IDENTITY_PASSWORD=my-signed-certificate-password)" "C:\Users\Public
\Documents\SQL Anywhere
    17\Samples\demo.db"
```

> ⚠ Caution
>
> The sample identity file is intended for use only for testing and development. Do not use it on a production system. Instead, replace it with your own certificate before deploying your application.

The database running on the database server must:

- Accept standard (database user ID and password) logins. If your database supports authentication mechanisms such as Windows integrated login, Kerberos, or PAM UA, then ensure that it also supports standard database logins.
  Have the COCKPIT_ROLE user-defined role defined. By default, in version 16 databases the COCKPIT_ROLE does not exist, so you must create it.

Your user credentials:

- must be granted the exercise rights to the COCKPIT_ROLE user-defined role. By default, the exercise rights to the COCKPIT_ROLE are not granted to any user, including the initial user, so you must grant it.
- need the following system privileges to access the full functionality of the Cockpit:
- - ACCESS DISK INFORMATION system privilege
  - BACKUP DATABASE system privilege
  - DROP CONNECTION system privilege
  - MONITOR system privilege
  - SERVER OPERATOR system privilege

Context

You can also start the Cockpit on a running database server using the sa_server_option system procedure.

Procedure

1. In SQL Central, connect to the database.
2. In the *Folders* pane, right-click the database server, and then click *Open SQL Anywhere Cockpit*.
3. Follow the instructions in the wizard.

   The Cockpit starts and a browser opens to the login page.

   The URL assigned to the Cockpit can also be retrieved by executing the following query: `SELECT PROPERTY ( 'CockpitURL' );`
4. Connect with your database user ID, password, and database name. The database name is not the Cockpit database name (the database that you just created when you performed step 2).

Next Steps

Continue to run the Cockpit in the background so that you can connect to it when needed.

Related Information

Cockpit Security [page 1429]
Upgrading Version 16 Databases to Use the SQL Anywhere Cockpit
Upgrading Version 17 Databases to Use the SQL Anywhere Cockpit
COCKPIT_ROLE User-defined Role [page 1544]
sa_server_option System Procedure

1.8.3.1.2 Starting and Connecting to the Cockpit (sa_server_option System Procedure)

Start the Cockpit on a running database server, and use it to view the current state of your database server.

Prerequisites

To stop and start the Cockpit, you need the EXECUTE system privilege for the sa_server_option system procedure and the SERVER OPERATOR system privilege.

The Cockpit runs on the database server that it monitors. You connect to the Cockpit using the credentials (database user ID and password) from a database that is running on the database server.

The database server must:

- Accept shared memory connections. For example, if the command to start the database server includes the -ec TLS database server option, then include the -es database server option.
- Have at least one version 16 or later database running on it.
- Have an HTTPS connection listener, so that the communication between the Cockpit and the web browser is encrypted. Otherwise, you risk a third party observing the communications between the Cockpit and web browser, and obtaining sensitive information including the database user name and password that you use to connect to the Cockpit.
  An HTTPS connection listener requires a certificate that must be either signed using a certificate that you create or signed by a certificate authority. Certificate authorities generally charge a fee to sign a certificate. You can create a signed certificate using the Create Certificate Creation utility (createcert) by creating a root certificate and then using it to create the signed certificate. Update your browser's list of trusted root certificates to include your root certificate file. This step is not required when you purchase the signed certificate from a certificate authority because browsers maintain a list of trusted certificates from common certificate authorities.
  Start an HTTPS connection listener when you start the database by specifying the -xs database server option and by specifying HTTPS as the protocol to start an HTTPS connection listener. For example:

```
dbsrv17 -xs "HTTPS(PORT=443;IDENTITY=c:\my-signed-certificate-identity-
file.id;IDENTITY_PASSWORD=my-signed-certificate-password)" "C:\Users\Public
\Documents\SQL Anywhere
    17\Samples\demo.db"
```

By default, an HTTPS connection listener is shared with all databases running on the database server. To restrict an HTTPS connection listener to only the Cockpit, use the DBN protocol option and specify *SQLACockpit* as the database-name. For example:

```
dbsrv17 -cdb "DBF=C:\temp\cockpitsettings.db" -xs
"HTTPS(PORT=443;DBN=SQLACockpit;IDENTITY=c:\my-signed-certificate-identity-
file.id;IDENTITY_PASSWORD=my-signed-certificate-password)" "C:\Users\Public
\Documents\SQL Anywhere
    17\Samples\demo.db"
```

> ⚠ Caution
>
> The sample identity file is intended for use only for testing and development. Do not use it on a production system. Instead, replace it with your own certificate before deploying your application.

The database running on the database server must:

- Accept standard (database user ID and password) logins. If your database supports authentication mechanisms such as Windows integrated login, Kerberos, or PAM UA, then ensure that it also supports standard database logins.
  Have the COCKPIT_ROLE user-defined role defined. By default, in version 16 databases the COCKPIT_ROLE does not exist, so you must create it.

Your user credentials:

- must be granted the exercise rights to the COCKPIT_ROLE user-defined role. By default, the exercise rights to the COCKPIT_ROLE are not granted to any user, including the initial user, so you must grant it.
- need the following system privileges to access the full functionality of the Cockpit:
- - ACCESS DISK INFORMATION system privilege
  - BACKUP DATABASE system privilege
  - DROP CONNECTION system privilege
  - MONITOR system privilege
  - SERVER OPERATOR system privilege

Procedure

1. Connect to the database, and then run the following sa_server_option system procedure to start the Cockpit:

   ```
   CALL sa_server_option('CockpitDB' 'START=ON;DBF=c:\temp\cockpitsettings.db');
   ```

2. Retrieve the URL to access the Cockpit by running the following query:

   ```
   SELECT PROPERTY ( 'CockpitURL' );
   ```

3. Open a browser and browse to the Cockpit URL.
4. Connect to the Cockpit by using your database user ID, password, and database name.

Next Steps

Continue to run the Cockpit in the background, so that you can connect to it when needed.

Related Information

Upgrading Version 16 Databases to Use the SQL Anywhere Cockpit
Upgrading Version 17 Databases to Use the SQL Anywhere Cockpit
COCKPIT_ROLE User-defined Role [page 1544]
-cdb Database Server Option [page 405]
Network Protocol Options [page 187]
sa_server_option System Procedure
List of Database Server Properties [page 900]
Cockpit Security [page 1429]

1.8.3.1.3 Stopping and Restarting the Cockpit (sa_server_option System Procedure)

Stop and restart the Cockpit that is running on a database server.

Prerequisites

The database server must have at least one version 16 or later database running on it. You must be a user of this database, who has the EXECUTE system privilege on the sa_server_option system procedure and the SERVER OPERATOR system privilege.

Procedure

1. Connect to a database that is running on same database server that the Cockpit is running on.
2. Run the following sa_server_option system procedure to stop the Cockpit:

   ```
   CALL sa_server_option('CockpitDB' 'START=OFF');
   ```
3. Complete the tasks that you need to perform while the Cockpit is stopped.
4. Restart the Cockpit by running the following sa_server_option system procedure:

   ```
   CALL sa_server_option('CockpitDB' 'START=ON');
   ```

1.8.3.1.4 Stopping and Restarting the Cockpit (SQL Central)

Stop and restart the Cockpit that is running on a database server.

Prerequisites

The database server must have at least one version 16 or later database running on it. You must be a user of this database, who has the EXECUTE system privilege on the sa_server_option system procedure and the SERVER OPERATOR system privilege.

Context

You can also stop and restart the Cockpit using the sa_server_option system procedure.

Procedure

1. In SQL Central, connect to a database that is running on same database server that the Cockpit is running on.
2. In the left pane, right-click the database server, and then click *Properties*.
3. Click *Options* in the *Properties* window.
4. Click *Stop Cockpit Now*.
5. Complete the tasks that you need to perform while the Cockpit is stopped.
6. Restart the Cockpit by clicking *Start Cockpit Now* from the same *Database Server Properties* window where you stopped it.

Related Information

sa_server_option System Procedure
-cdb Database Server Option [page 405]

1.8.3.1.5 Setting up the Cockpit in the SAP DB Control Center

Register the Cockpit with the SAP DB Control Center to centrally monitor the health of your database server along with your other enterprise systems.

Prerequisites

The Cockpit must be running on a database server that has at least one other database running on it. You must be a user of that database and have the COCKPIT_ROLE user-defined role.

The SAP DB Control Center must be running. You must be an SAP DB Control Center user who can add a cockpit system to the SAP DB Control Center.

Context

The SAP DB Control Center connects to the Cockpit to retrieve and display summarized information about the health of the database server. To connect to the Cockpit, the SAP DB Control Center requires a user name and password. Create and store these credentials within the Cockpit. The user that you create can only connect to the Cockpit from the SAP DB Control Center to retrieve summarized information.

Procedure

1. Start the Cockpit on a database server and create an HTTPS connection listener that can only be used by the Cockpit. Start a database on the same database server.

   The following command starts the sample database and the Cockpit. It uses the DBN parameter with the -xs database server option to create an HTTPS connection listener that can only be used by the Cockpit.

   ```
   dbsrv17 -n my_cockpitserver -gk all -xs
   "HTTPS(PORT=443;DBN=SQLACockpit;IDENTITY=C:\Users\Public\Documents\SQL Anywhere
       17\Samples\Certificates\rsaserver.id;IDENTITY_PASSWORD=test)" -cdb "DBF=C:
   \temp\cockpitsettings.db"  "C:\Users\Public\Documents\SQL Anywhere
       17\Samples\demo.db"
   ```

2. Connect to the Cockpit.
   a. Open a browser and browse to the Cockpit URL.

      The URL assigned to the Cockpit can be retrieved by running the following query: `SELECT PROPERTY ( 'CockpitURL' );`.
   b. Connect to the Cockpit by using your database user ID, password, and database name.
3. Create credentials for the SAP DB Control Center to use to retrieve and display summarized information about the health of the database server that the Cockpit monitors.

a. On the *Home* tab, click *Show details* for the database server, and then click *SAP DB Control Center settings*.
b. Click *Set the credentials for the SAP DB Control Center*, and create a user name and password for the SAP DB Control Center to use to connect to the Cockpit.

4. Log in to the SAP DB Control Center.
5. Follow the instructions in the SAP DB Control Center documentation to register the Cockpit. Use the user name and password that you created for the SAP DB Control Center user to register the Cockpit.

Related Information

User and Database Security [page 1525]
COCKPIT_ROLE User-defined Role [page 1544]
Starting and Connecting to the Cockpit(SQL Central) [page 1421]
sa_server_option System Procedure
-cdb Database Server Option [page 405]

1.8.3.1.6 Enabling the Cockpit to Send Alert Emails

Configure the Cockpit to send email notifications to users when alerts occur.

Prerequisites

A database must be running on the database server, and you must have a user that has the COCKPIT_ROLE user-defined role and the following system privileges:

- ACCESS DISK INFORMATION system privilege
- BACKUP DATABASE system privilege
- DROP CONNECTION system privilege
- MONITOR system privilege
- SERVER OPERATOR system privilege

You must know your email server settings.

Procedure

1. On the *MONITORING* tab, click *Configure Alerts* > *Email Notifications*, and then select *Send alert notifications by email*.
2. Choose one of the following email protocols and specify the required settings:

Option	Action
SMTP server	Specify the email server, port, sender name, and sender address.
MAPI server	Specify a user ID and password.

3. Click *Send Test Email* to verify your email settings.
4. Open the *Addresses* tab, click *Add*, and specify the email addresses to receive the email notifications.
5. Open the *Options* tab.
   a. Configure the rules for sending email notifications.
   b. In the *Use this external URL or IP address for the Cockpit* field, specify the *external* name or IP address of the computer that the Cockpit runs on. This external name or IP address is used in the links provided in the notification emails.
6. Click *OK*.

Results

Notification emails are sent to the specified addresses when an alert occurs that meets or exceeds the specified criteria.

1.8.3.1.7 Cockpit Security

There are many security features that you can use to secure the Cockpit.

Connect to the Cockpit by using the credentials of a user from one of the databases running on the database server. At minimum, this database must be version 16 or later, and it must have the COCKPIT_ROLE user-defined role defined. Your credentials must also have exercise rights to the COCKPIT_ROLE user-defined role.

By default the exercise rights to the COCKPIT_ROLE are not granted to any user, including the initial user. You must grant its exercise rights to users. In version 16 databases, create the COCKPIT_ROLE and then grant it to users.

The Cockpit does not support databases that use the legacy definer security model.

Prevent Users from Connecting to the Cockpit

To prevent database users from connecting to the Cockpit, do not grant them exercise rights to the COCKPIT_ROLE user-defined role. To prevent all users of a database from being able to access the Cockpit, delete the COCKPIT_ROLE user-defined role.

Restrict the Actions of a User in the Cockpit

To perform any authorized task on a database, connect as a user of that database with the roles and privileges required to perform the task. A user who is connected to the Cockpit cannot do anything or see anything using the Cockpit that they could not do or see by using SQL statements on a normal database connection with the credentials they provide.

In newly created databases, the COCKPIT_ROLE user-defined role is populated with the system privileges required to perform any task in the Cockpit. Review this list of privileges and revoke privileges as needed to maintain your security model. For example, John and Jane are both users of the MySample database, and they both must connect to the Cockpit. According to the database's security model only Jane is allowed to back up the database. Remove remove the BACKUP DATABASE system privilege from the COCKPIT_ROLE, and then grant the COCKPIT_ROLE role to both John and Jane. Then you could grant Jane the BACKUP DATABASE system privilege.

Prevent Users from Starting the Cockpit

Any user who can start a database server can start the Cockpit when the database server starts.

Use the -sf database server option to prevent a user from starting the Cockpit on a running database server.

Encrypt Communication Between the Cockpit and Your Browser

Start an HTTPS connection listener to encrypt the communication between the Cockpit and the web browser. If you do not start an HTTPS connection listener, then the Cockpit starts an unsecured HTTP connection listener. With an unsecured HTTP connection listener, a third party could observe communications between the Cockpit and web browser and obtain sensitive information, including the database user name and password that are used to connect to the Cockpit. Specify the -xs database server option to create an HTTPS connection listener.

Example

The following command starts the sample database and the Cockpit. It uses the DBN parameter to create an HTTPS connection listener that can only be used by the Cockpit.

```
dbsrv17 -cdb "DBF=C:\temp\cockpitsettings.db" -xs
"HTTPS(PORT=443;DBN=SQLACockpit;IDENTITY=c:\my-signed-certificate-identity-
file.id;IDENTITY_PASSWORD=my-signed-certificate-password)" "C:\Users\Public\Documents
\SQL Anywhere
    17\Samples\demo.db"
```

Related Information

COCKPIT_ROLE User-defined Role [page 1544]
User and Database Security [page 1525]
-xs Database Server Option [page 529]

1.8.3.1.8 Cockpit Tutorial

Learn how to securely start the Cockpit, and monitor the availability, capacity, and performance of a database server.

Prerequisites

This tutorial uses the sample database. Verify that the sample database (*demo.db*) has the COCKPIT_ROLE user-defined role, and that the exercise rights for this user-defined role, as well as the exercise rights for the following system privileges are granted to your user:

- MONITOR system privilege
- DROP CONNECTION system privilege
- BACKUP DATABASE system privilege
- SERVER OPERATOR system privilege

You need the EXECUTE system privilege for the sa_server_option system procedure and the SERVER OPERATOR system privilege to complete other steps in this tutorial.

Context

In this tutorial, you connect to the Cockpit by specifying the credentials of a user from the sample database that is running on the same database server as the Cockpit.

In the sample database, the database administrator user has both the administrator and exercise rights granted for the COCKPIT_ROLE user-defined role. But, in newly created databases, only the administrator rights are granted to the database administrator, so you must grant the COCKPIT_ROLE user-defined role to any user that needs to connect to the Cockpit.

In this section:

Lesson 1: Setting up the Cockpit Securely Using an HTTPS Connection Listener [page 1432]
Secure the communication between the Cockpit and your browser by using an HTTPS connection listener with a signed certificate.

Lesson 2: Viewing an Alert in the Cockpit (SQL Central) [page 1434]
Experiment with the SQL Anywhere Cockpit user interface, and test an alert.

1.8.3.1.8.1 Lesson 1: Setting up the Cockpit Securely Using an HTTPS Connection Listener

Secure the communication between the Cockpit and your browser by using an HTTPS connection listener with a signed certificate.

Prerequisites

You must have the roles and privileges listed at the beginning of this tutorial.

You must know the name of the host (computer name) of the computer where the Cockpit will run.

Context

An HTTPS connection listener requires a signed certificate that was either purchased from a certificate authority (CA), or created using the Certificate Creation utility (createcert).

This tutorial describes how to create a signed certificate by using Certificate Creation utility (createcert) and how to use the certificate with the Cockpit. When you create the signed certificate, you must update your browser's list of trusted certificates to include the root certificate file that was used to sign your certificate. This step is not required when you purchase the signed certificate from a certificate authority because browsers maintain a list of trusted certificates from common signing authorities.

Procedure

1. Create the signed certificate.
   a. Run the Certificate Creation utility (createcert) to create a root certificate that has signing authority

      Run `createcert` and specify the following options:

```
Enter RSA key length (512-16384): 2048
Generating key pair...
Country Code: ca
State/Province: on
Locality: waterloo
Organization: mycompany
Organizational Unit: eng
Common Name:myrootcommonname
Enter file path of signer's certificate:
Certificate will be a self-signed root
Serial number [generate GUID]:
Generated serial number: a5cf457769724934aed4d584d0d153a9
Certificate valid for how many years (1-100): 1
Certificate Authority (Y/N) [N]: y
1.  Digital Signature
2.  Nonrepudiation
3.  Key Encipherment
4.  Data Encipherment
```

```
5.  Key Agreement
6.  Certificate Signing
7.  CRL Signing
8.  Encipher Only
9.  Decipher Only
Key Usage [6,7]:
Enter file path to save certificate: myroot.crt
Enter file path to save private key: myroot.key
Enter password to protect private key: myrootpassword
Enter file path to save identity: myroot.id
```

b. Use the root certificate, `myroot.crt`, to create the signed certificate that database server uses to secure the HTTPS connection listener.

   Run `createcert` and specify the following options:

```
Enter RSA key length (512-16384): 2048
Generating key pair...
Country Code: ca
State/Province: on
Locality: waterloo
Organization: mycompany
Organizational Unit: eng
Common Name: Specify the host name of the Cockpit
Enter file path of signer's certificate: myroot.crt
Enter file path of signer's private key: myroot.key
Enter password for signer's private key: myrootpassword
Serial number [generate GUID]:
Generated serial number: 78ea0f2e763247d49f3b5bc05a0d3dde
Certificate valid for how many years (1-100): 1
Certificate Authority (Y/N) [N]: n
1.  Digital Signature
2.  Nonrepudiation
3.  Key Encipherment
4.  Data Encipherment
5.  Key Agreement
6.  Certificate Signing
7.  CRL Signing
8.  Encipher Only
9.  Decipher Only
Key Usage [1,3,4,5]:
Enter file path to save certificate: myservercert.crt
Enter file path to save private key: myservercert.key
Enter password to protect private key: myservercertpassword
Enter file path to save identity: myservercert.id
```

   The value for the *Common name* parameter must match that of the host (the name of the computer), on which the Cockpit will run.

2. Add the root certificate, `myroot.crt`, to the list of trusted root certificate authorities for your browser. See your browser's documentation for instructions. Repeat this step for each browser and on each computer that you want to use to access the Cockpit.
3. Start the Cockpit and the sample database, *demo.db*, on the same database server.

```
dbsrv17 -cdb "DBF=C:\temp\cockpitsettings.db" -xs
"HTTPS(PORT=443;IDENTITY=myservercert.id;IDENTITY_PASSWORD=myservercertpasswor
d)" "C:\Users\Public\Documents\SQL Anywhere
    17\Samples\demo.db"
```

   The -cdb database server option starts the Cockpit and the -xs database server option with the HTTPS protocol starts an HTTPS connection listener.

By default, an HTTPS connection listener is shared with all databases running on the database server. To restrict an HTTPS connection listener to only the Cockpit, use the DBN protocol option and specify *SQLACockpit* as the database name.

The Cockpit starts.

4. Open the Cockpit.
   a. In SQL Central, connect to the sample database.
   b. In the *Folders* pane, right-click the database server, and then click *Start SQL Anywhere Cockpit*.

   A browser opens to the Cockpit connection page.

   The URLs to access the Cockpit appear in the *Database Server Messages* window. The URLs can also be obtained by querying the CockpitURL database server property: `SELECT PROPERTY ( 'CockpitURL' ).`
5. Create a user with the COCKPIT_ROLE role that can connect to the Cockpit:.

   ```
   CREATE USER user1 IDENTIFIED BY 'password';
   GRANT ROLE COCKPIT_ROLE TO user1;
   ```
6. Connect to the Cockpit as user1.

Next Steps

Proceed to the next lesson.

Related Information

Certificate Creation Utility (createcert) [page 1130]
Network Protocol Options [page 187]

1.8.3.1.8.2 Lesson 2: Viewing an Alert in the Cockpit (SQL Central)

Experiment with the SQL Anywhere Cockpit user interface, and test an alert.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Procedure

1. Look up a database server property.
   a. In the Cockpit, navigate to the *Home* page.
   b. Click *Show Details* for the database server to find the command line used to start the database server.
   c. On the *Properties* page, type **`CommandLine`** in the filter box in the upper right corner.

      The CommandLine property and its value appears in the list.
   d. Close the window.
2. Configure an alert threshold.
   a. Open the *MONITORING* page.
   b. Click *Configure Alerts*.
   c. Adjust the threshold for the *Raise a low alert when there are* X *or more connections to the server* alert to **`1`**.
   d. Click *OK*.

   The conditions for this alert are true, so the alert is triggered. The alert appears in the table on the *MONITORING* page. It takes up to 20 seconds for the alert to appear.
3. Click the name of the Alert to view its details, and then close the *Alert* window.
4. Stop the Cockpit.
   a. In SQL Central, right-click the database server and then click *Properties*.
   b. Click *Options*.
   c. Click *Stop Cockpit Now*.

Results

You have completed the tutorial on using the Cockpit.

Related Information

COCKPIT_ROLE User-defined Role [page 1544]
COCKPIT_ROLE User-defined Role [page 1544]
sa_server_option System Procedure

1.8.3.2 Tips for Improving Performance

There are several tips to improve the performance of the software.

Server-Related Performance Tips

- Tip: Use the cache to improve performance
- Tip: Check for concurrency issues
- Tip: Choose the optimizer goal
- Tip: Update column statistics
- Tip: Use indexes effectively
- Tip: Improve index performance
- Tip: Optimize for mixed or OLAP workload
- Tip: Use strategic sorting of query results
- Tip: Specify the correct cursor type
- Tip: Supply explicit selectivity estimates sparingly
- Tip: Use materialized views to improve query performance
- Tip: Use the WITH EXPRESS CHECK option when validating tables
- Tip: Use work tables in query processing (use All-rows optimization goal)

Application-Related Performance Tips

- Tip: Build efficient SQL queries
- Tip: Monitor query performance
- Tip: Reduce expensive user-defined functions
- Tip: Turn off autocommit mode
- Tip: Use in-memory mode

Database-Related Performance Tips

- Tip: Always use a transaction log
- Tip: Use delayed commits
- Tip: Collect statistics on small tables
- Tip: Reduce fragmentation
- Tip: Normalize your table structure
- Tip: Minimize cascading referential actions
- Tip: Declare constraints

- Tip: Place different files on different devices
- Tip: Rebuild your database
- Tip: Use keys to improve query performance
- Tip: Reduce primary key width
- Tip: Reduce table widths
- Tip: Review the order of columns in tables
- Tip: Replace expensive triggers
- Tip: Use an appropriate page size
- Tip: Use AUTOINCREMENT to create primary keys
- Tip: Use appropriate data types
- Tip: Use bulk operations methods
- Tip: Use resource governors

Communication-Related Performance Tips

- Tip: Reduce requests between client and server
- Tip: Use compression carefully
- Tip: Change packet size to improve performance

In this section:

Tip: Use the Cache to Improve Performance [page 1440]
The cache is an area of memory used by the database server to store database pages for repeated fast access.

Tip: Check for Concurrency Issues [page 1447]
Concurrent access of information stored in the database by other transactions is prevented by locks that maintain the reliability of information.

Tip: Identify the Cause of Slow Statements [page 1447]
The **statement performance summary** feature returns execution times for statements when troubleshooting slow statement performance. The *SQL Anywhere Profiler* uses this feature to display the statement performance summary results.

Tip: Choose the Optimizer Goal [page 1448]
You can choose whether to optimize query processing to return the first row quickly, or to minimize the cost of returning the complete result set.

Tip: Update Column Statistics [page 1448]
To continually improve optimizer performance, the database server automatically updates column statistics during the processing of any SELECT, INSERT, UPDATE, or DELETE statement.

Tip: Use Indexes Effectively [page 1450]
The database server can evaluate search conditions with the aid of indexes.

Tip: Improve Index Performance [page 1451]
You can improve your index performance by reorganizing composite indexes or increasing the page size.

Tip: Optimize for Mixed or OLAP Workload [page 1451]

Use the optimization_workload option to specify optimization preference.

Tip: Use Strategic Sorting of Query Results [page 1451]
Use strategic sorting of query results to reduce the amount of unnecessary sorting of data.

Tip: Specify the Correct Cursor Type [page 1451]
Specify the correct cursor type to improve performance.

Tip: Supply Explicit Selectivity Estimates Sparingly [page 1452]
Although user defined estimates can improve performance, avoid supplying explicit user-defined estimates in statements that are to be used on an ongoing basis.

Tip: Use Materialized Views to Improve Query Performance [page 1452]
Consider using materialized views for frequently executed, expensive queries, such as those involving intensive aggregation and join operations.

Tip: Use the WITH EXPRESS CHECK Option When Validating Tables [page 1456]
The WITH EXPRESS CHECK option with the VALIDATE TABLE statement, or the -fx option with the Validation utility (dbvalid) can significantly increase the speed at which your tables validate

Tip: Use Work Tables in Query Processing (Use All-rows Optimization Goal) [page 1456]
Work tables are materialized temporary result sets that are created during the execution of a query.

Tip: Build Efficient SQL Queries [page 1457]
To improve query processing performance, consider building more efficient queries.

Tip: Monitor Query Performance [page 1462]
Several tools for testing the performance of queries are provided.

Tip: Reduce Expensive User-defined Functions [page 1463]
Reducing expensive user-defined functions in queries where they have to be executed many times can improve performance.

Tip: Turn Off Autocommit Mode [page 1463]
Instead of running in autocommit mode, consider grouping your SQL statements so each group performs one logical task.

Tip: Use In-memory Mode [page 1464]
If your application can tolerate the loss of committed transactions after the most recent checkpoint, then your application may benefit from using in-memory mode.

Tip: Check the Lagtime Mirror Option Setting (Mirrored Systems Only) [page 1464]
The lagtime mirror option can impact performance on mirrored systems.

Tip: Always Use a Transaction Log [page 1464]
Using a transaction log can provide data protection, and can improve the performance of the database server.

Tip: Use Delayed Commits [page 1465]
Delayed commits determine the rate of response a database provides to a COMMIT statement.

Tip: Collect Statistics on Small Tables [page 1465]
By default, the database server creates statistics for tables with more than five rows. To create statistics for a table with less than five rows, use the CREATE STATISTICS statement.

Tip: Reduce Fragmentation [page 1465]
Fragmentation occurs as you make changes to your database, and performance can suffer if your files, tables, or indexes are excessively fragmented.

Tip: Normalize Your Table Structure [page 1470]

Database tables may contain multiple copies of the same information (for example, a column that is repeated in several tables), and your table may need to be normalized.

Tip: Minimize Cascading Referential Actions [page 1471]
Cascading referential actions are costly because they cause updates to multiple tables for every transaction. This can affect performance.

Tip: Declare Constraints [page 1471]
Declaring primary key and foreign key relationships and their constraints can improve performance.

Tip: Place Different Files on Different Devices [page 1471]
You can improve database performance by putting different database files on different physical devices or drives.

Tip: Rebuild Your Database [page 1472]
Rebuilding your database can improve performance.

Tip: Use Keys to Improve Query Performance [page 1472]
Primary keys and foreign keys can also improve database performance.

Tip: Reduce Primary Key Width [page 1473]
Reducing the number of columns in your primary keys can improve performance.

Tip: Reduce Table Widths [page 1473]
If you have wide tables, and find performance slow consider further normalizing your tables to reduce the number of columns.

Tip: Review the Order of Columns in Tables [page 1473]
The order of the columns in a table affects performance.

Tip: Replace Expensive Triggers [page 1473]
Evaluate the use of triggers to see if some of the triggers could be replaced by features available in the database server.

Tip: Use an Appropriate Page Size [page 1474]
The database page size can affect the performance of your database.

Tip: Use AUTOINCREMENT to Create Primary Keys [page 1475]
Using the AUTOINCREMENT feature to generate primary key values is faster than other methods because the value is generated by the database server.

Tip: Use Appropriate Data Types [page 1475]
You can improve performance by using the appropriate data type for your data.

Tip: Use Bulk Operations Methods [page 1475]
If you load large amounts of information into your database, you can benefit from the special tools provided for these tasks.

Tip: Use Resource Governors [page 1476]
Resource governors can be used to improve performance.

Tip: Reduce Requests Between Client and Server [page 1476]
Use the LazyClose and PrefetchOnOpen network connection parameters to reduce the number of requests between the client and server.

Tip: Use Compression Carefully [page 1477]
Enabling compression for one connection or all connections, and adjusting the minimum size limit at which packets are compressed can offer significant improvements to performance.

Tip: Change Packet Size to Improve Performance [page 1477]

In some cases, increasing the packet size can improve performance.

1.8.3.2.1 Tip: Use the Cache to Improve Performance

The cache is an area of memory used by the database server to store database pages for repeated fast access.

The more pages that are accessible in the cache, the fewer times the database server needs to read data from disk, which is a slower operation. Cache size is therefore often a key factor in performance.

Dynamic cache sizing, which tunes the cache size appropriately and automatically by monitoring the system as a whole, is supported. You can also use the -c option on the database server command line when the database is started to control the size of the database cache.

The database server messages window displays the size of the cache at startup, but you can also use the following statement to obtain the current size of the cache:

```
SELECT PROPERTY( 'CurrentCacheSize' );
```

Encrypted databases must have sufficient cache to minimize I/O operations because these operations are more expensive on encrypted databases than on unencrypted databases since encryption and/or decryption must be performed for each operation.

In this section:

1.8.3.2.1.1 Cache and the Memory Governor

The database server uses the cache (buffer pool) to temporarily store images of database pages in memory.

These pages are typically table pages and index pages, although there are several other types of physical pages stored in a database. In addition to these pages, the database server uses the cache for two other pools of memory. One of these pools is the virtual memory used for database server data structures, such as those that represent connections, statements, and cursors. The second pool consists of cache pages that are used as virtual storage for query memory.

Query execution requires memory to operate efficiently. The database server uses a memory governor to decide how much query memory each statement can use for query execution. The memory governor is responsible for allocating a pool of query memory to statements to provide efficient execution of the workload.

The memory governor grants individual statements a selected number of pages that the statement can use for memory-intensive query processing. Memory in the query memory pool is still available for other purposes (such as buffering table or index pages) until the query processor uses the pages. Memory-intensive query processing that uses query memory includes all hash-based operators, such as hash distinct, hash group by, and hash join, and sorting and window operators.

Use the following settings, operators, and statistics to understand, and control, how the memory governor uses the cache:

QueryMemMaxUseful graphical plan operator

When a statement begins executing, the memory governor uses the optimizer's estimates to determine how much memory would be useful to the statement. This estimate appears in the graphical plan as QueryMemMaxUseful.

QueryMemActiveMax server property

The memory governor limits the number of memory-intensive requests that can execute concurrently. This maximum value is selected based on the performance characteristics of the computer running the database server, and the limit is shown with the server property QueryMemActiveMax.

QueryMemActiveEst Performance Monitor statistic

The memory governor maintains a running estimate of the number of concurrent memory intensive requests, and this estimate is available as the database server property and Performance Monitor statistic QueryMemActiveEst.

query_mem_timeout database option

If a memory-intensive statement begins executing and there are already the maximum number of concurrent memory-intensive requests executing, then incoming statements wait for one of the existing requests to release its allocated memory. The query_mem_timeout database option controls how long the incoming request waits for a memory grant. With the default setting of -1, the request waits for a database server-defined period of time. If no memory grant is available after waiting, then the statement's access plan is executed with a small amount of memory, which could cause it to perform slowly, possibly with a low-memory execution strategy if one exists for memory-intensive physical operators in that plan.

QueryMemGrantWaiting server property and Performance Monitor statistic

The database server property and Performance Monitor statistic QueryMemGrantWaiting shows the current number of requests that are waiting for a memory request to be granted.

QueryMemGrantWaited server property and Performance Monitor statistic

The database server property and Performance Monitor statistic QueryMemGrantWaited shows the total number of times that a request had to wait before a memory request was granted.

QueryMemNeedsGrant graphical plan operator

In the graphical plan, QueryMemNeedsGrant shows whether the memory governor considers the request to be a simple request (no memory grant needed) or memory intensive (a memory grant is needed). If the memory governor classifies a request as not needing a memory grant, then the request begins executing immediately. Otherwise, the request asks to use a portion of the query memory pool.

QueryMemLikelyGrant graphical plan operator

In the graphical plan, **QueryMemLikelyGrant** shows an estimate of how many pages are likely to be granted to the request for execution.

1.8.3.2.1.2 Cache and the Optimizer

The database server associates a higher cost with execution plans that require additional buffer cache overhead.

Reserving extra memory, for example to hold the contents of a cursor, may be expensive. If the cache is full, one or more pages may have to be written to disk to make room for new pages. Some pages may need to be re-read to complete a subsequent operation. This cost discourages the optimizer from choosing plans that use work tables. However, the optimizer is careful to use memory where it improves performance. For example, it caches the results of subqueries when they are needed repeatedly during query processing.

1.8.3.2.1.3 Limit Cache Memory Use

The initial, minimum, and maximum cache sizes are controllable from the database server command line.

Initial cache size

You can specify the initial cache size for the database server by using the -c database server option. If you do not specify the -c option, the database server calculates the initial cache allocation.

Maximum cache size

You can control the maximum cache size by specifying the database server -ch option. The default is based on a heuristic that depends on the physical memory in your computer. On non-Unix computers, this is approximately the lower of the maximum cache size and 90% of the physical memory of the computer. On Unix, the default maximum cache size is calculated as follows:

- On 32-bit Unix platforms, it is the lesser of 90% of total physical memory or 1,834,880 KB.
- On 64-bit Unix platforms, it is the lesser of 90% of total physical memory and 8,589,672,320 KB.

Minimum cache size

You can control the minimum cache size by specifying the database server -cl server option. By default, the minimum cache size is the same as the initial cache size.

If you specify the -c server option without -cl, then the minimum cache size is set to the initial cache size specified by the -c server option.

If you do not set the -c or -cl server options, the minimum cache size is set to a very low hard-coded constant value, so that the cache can shrink if necessary.

If you attempt to set your initial or minimum cache sizes to a value that is less than one eighth of the maximum cache size, the initial and minimum cache sizes are automatically increased by an amount computed from the maximum cache size.

You can also disable dynamic cache sizing by using the -ca 0 server option.

The following database server properties return information about the database server cache:

CurrentCacheSize

Returns the current cache size, in kilobytes.

MinCacheSize

Returns the minimum allowed cache size, in kilobytes.

MaxCacheSize

Returns the maximum allowed cache size, in kilobytes.

PeakCacheSize

Returns the largest value the cache has reached in the current session, in kilobytes.

1.8.3.2.1.4 Dynamic Cache Sizing

With **dynamic cache sizing**, the cache grows when more memory is made available to the database server and shrinks when cache memory is required by other applications.

The effectiveness of dynamic cache sizing varies depending on the operating system and the amount of available physical memory.

Typically, the cache requirements are assessed by dynamic cache sizing once per minute. However, when a new database is started or when a file grows significantly, the assessment frequency may increase to once every five seconds for thirty seconds. After the initial thirty second period, the assessment frequency returns to once per minute. File growth is considered significant if it is one eighth greater than the last growth that triggered an increase in the assessment frequency or one eighth greater than when the database started.

With dynamic cache sizing you do not need to explicitly configure the database cache.

If you attempt to set your initial or minimum cache sizes to a value that is less than one eighth of the maximum cache size, the initial and minimum cache sizes are automatically increased by an amount computed from the maximum cache size.

In this section:

Dynamic Cache Sizing on Microsoft Windows [page 1444]
On Windows, the database server evaluates cache and operating statistics once per minute and computes an optimum cache size.

Dynamic Cache Sizing on UNIX/Linux [page 1444]
On UNIX and Linux, the database server uses swap space and memory to manage the cache size.

1.8.3.2.1.4.1 Dynamic Cache Sizing on Microsoft Windows

On Windows, the database server evaluates cache and operating statistics once per minute and computes an optimum cache size.

The database server computes a target cache size that uses all physical memory currently not in use, except for approximately 5 MB that is to be left free for system use. The target cache size is never smaller than the specified or implicit minimum cache size. The target cache size never exceeds the specified or implicit maximum cache size, or the sum of the sizes of all open database and temporary files plus the size of the main heap.

To avoid cache size oscillations, the database server increases the cache size incrementally. Rather than immediately adjusting the cache size to the target value, each adjustment modifies the cache size by 75% of the difference between the current and target cache size.

1.8.3.2.1.4.2 Dynamic Cache Sizing on UNIX/Linux

On UNIX and Linux, the database server uses swap space and memory to manage the cache size.

The swap space is a system-wide resource on most UNIX and Linux operating systems. The sum of memory and swap space is called the **system resources**. See your operating system documentation for details.

On startup, the database allocates the specified maximum cache size from the system resources. It loads some of this into memory (the initial cache size) and keeps the remainder as swap space.

The total amount of system resources used by the database server is constant until the database server shuts down, but the proportion loaded into memory changes. Each minute, the database server evaluates cache and operating statistics. If the database server is busy and demanding of memory, it may move cache pages from swap space into memory. If the other processes in the system require memory, the database server may move cache pages out from memory to swap space.

Initial Cache Size

By default, the initial cache size is assigned using a heuristic based on the available system resources. The initial cache size is always less than 1.1 times the total database size.

If the initial cache size is greater than three quarters of the available system resources, the database server exits with an error indicating there is not enough memory.

You can change the initial cache size using the -c option.

Maximum Cache Size

The maximum cache must be less than the available system resources on the computer. By default, the maximum cache size is assigned using a heuristic based on the available system resources and the total

physical memory on the computer. The cache size never exceeds the specified or implicit maximum cache size, or the sum of the sizes of all open database and temporary files plus the size of the main heap.

If you specify a maximum cache size greater than the available system resources, the database server exits with an error indicating there is not enough memory. If you specify a maximum cache size greater than the available memory, the database server warns of performance degradation, but does not exit.

The database server allocates all the *maximum* cache size from the system resources, and does not relinquish it until the database server exits. Choose a maximum cache size that gives good performance while leaving space for other applications. The formula for the default maximum cache size is a heuristic that attempts to achieve this balance. Tune the value only if the default value is not appropriate on your system.

You can use the -ch server option to set the maximum cache size, and limit automatic cache growth.

Minimum Cache Size

If the -c option is specified, the minimum cache size is the same as the initial cache size. If no -c option is specified, the minimum cache size on UNIX and Linux is 8 MB.

You can use the -cl server option to adjust the minimum cache size.

If you attempt to set your initial or minimum cache sizes to a value that is less than one eighth of the maximum cache size, the initial and minimum cache sizes are automatically increased by an amount computed from the maximum cache size.

1.8.3.2.1.5 Statistics That Monitor Cache Size

Several statistics are included in the Windows Performance Monitor and the database's property functions.

CurrentCacheSize

The current cache size in kilobytes.

MinCacheSize

The minimum allowed cache size in kilobytes.

MaxCacheSize

The maximum allowed cache size in kilobytes.

PeakCacheSize

The peak cache size in kilobytes.

1.8.3.2.1.6 Cache Warming

Cache warming reduces the execution times of queries executed against a database.

To warm the cache, the database server checks whether the database contains a previously recorded collection of pages. If it does, the database server loads the corresponding pages into the cache. The database

can still process requests while the cache is loading pages, but warming may stop if a significant amount of I/O activity is detected in the database. Cache warming stops in this case to avoid performance degradation of queries that access pages that are not contained in the set of pages being reloaded into the cache.

Cache Warming After Restart

This is done by preloading the database server's cache with database pages that were referenced the last time the database was started. Warming the cache can improve performance **only** when the same query or similar queries are executed against a database each time it is started. However, if the statements executed at database startup are different than those executed the last time the database was started with cache collection on, cache warming does not improve performance.

You control the cache warming settings on the database server command line. There are two activities that can take place when a database is started and cache warming is turned on: collection of database pages and cache reloading (warming).

Collection of referenced database pages is controlled by the -cc database server option, and is turned on by default. When database page collection is turned on, the database server keeps track of every database page that is requested from database startup until one of the following occurs: the maximum number of pages has been collected (the value is based on cache size and database size), the collection rate falls below the minimum threshold value, or the database is shut down. The database server controls the maximum number of pages and the collection threshold. Once collection completes, the referenced pages are recorded in the database so they can be used to warm the cache the next time the database is started.

Cache warming (reloading) is turned on by default, and is controlled by the -cr database server option. You can specify the -cv option if you want messages about cache warming to appear in the database server messages window.

Cache Warming to a Steady State

Warming the cache to a steady state is done by saving the contents of the cache when the database is running in a steady state and then restoring the cache to that state when necessary. Cache warming to a steady state can improve performance in a situation where you have good performance during a steady state, but then find that certain queries and operations are slow until the cache contents have stabilized. During this period, performance can be slow, partly because each query may not be optimized with benefits to later queries in mind, resulting in the selection of a more expensive plan that uses pages needed by later queries. For example, a database administrator can save the contents of the cache before restarting a database, and restore the cached contents after the database has been restarted to minimize the disruption caused by a normal restart.

Cache warming to a steady state is off by default and is controlled by the ALTER DATABASE cache warming syntax or by the sp_db_cache_contents system procedure and the sp_read_db_pages system procedure. A database administrator must explicitly record the state of the cache and restore the pages that were present in that stored state.

1.8.3.2.2 Tip: Check for Concurrency Issues

Concurrent access of information stored in the database by other transactions is prevented by locks that maintain the reliability of information.

When the database server processes a transaction, it can lock one or more table rows. They also improve the accuracy of result queries by identifying information that is in the process of being updated.

When the database server processes a transaction, it can lock one or more table rows. The locks maintain the reliability of information stored in the database by preventing concurrent access by other transactions. They also improve the accuracy of result queries by identifying information that is in the process of being updated.

The database server places these locks automatically and needs no explicit instruction. It holds all the locks acquired by a transaction until the transaction is completed. The transaction that has access to the row is said to hold the lock. Depending on the type of lock, other transactions may have limited access to the locked row, or none at all.

Performance can be compromised if a row or rows are frequently accessed by several users simultaneously. If you suspect locking problems, consider using the sa_locks procedure to obtain information about locks in the database.

If lock issues are identified, information about the connection processes involved can be found using the AppInfo connection property.

1.8.3.2.3 Tip: Identify the Cause of Slow Statements

The **statement performance summary** feature returns execution times for statements when troubleshooting slow statement performance. The *SQL Anywhere Profiler* uses this feature to display the statement performance summary results.

The statement performance summary feature uses the sp_top_k_statements and sp_find_top_statements system procedures to report the statement/plan combinations that take the longest time to run.

Use the statement performance summary feature to answer questions like:

- Is this statement running slower today than it was before?
- Is the amount of data being returned or modified by the statement the same today as it was yesterday?
- Has the execution plan for the statement changed?
- Has one execution plan been used more yesterday than today?
- Is the maximum runtime for the statement much higher than the average runtime?
- Is the maximum/average runtime for the statement for one plan very different from the maximum/average runtime for the other statement? If so, do invocations with one plan process or return more rows than those with another plan?

The SQL for statements with a maximum runtime of 0.005 seconds or higher can be found in the GTSYSPERFCACHESTMT system view. The graphical plan for every statement or plan with the maximum runtime of 0.05 seconds can be found in the GTSYSPERFCACHEPLAN system view.

To disable statement performance summary collection on a single database, set the CollectStmtPerfStats option of the sa_db_option system procedure. Setting this option requires MONITOR privileges.

The statement performance summary feature only maintains information for a maximum of 10,000 statements with the highest maximum runtimes.

1.8.3.2.4 Tip: Choose the Optimizer Goal

You can choose whether to optimize query processing to return the first row quickly, or to minimize the cost of returning the complete result set.

The optimization_goal option controls whether the database server optimizes SQL statements for response time (First-row) or for total resource consumption (All-rows).

If the option is set to First-row, the database server chooses an access plan that is intended to reduce the time to fetch the first row of the query result, possibly at the expense of total retrieval time. In particular, the optimizer typically avoids, if possible, access plans that require the materialization of results to reduce the time to return the first row. With this setting, for example, the optimizer favors access plans that use an index to satisfy a query's ORDER BY clause, rather than plans that require an explicit sorting operation.

The optimization goal used by the optimizer for a particular statement is decided using these rules:

- If the main query block has a table in the FROM clause with the table hint set to FASTFIRSTROW, then the statement is optimized using the First-row optimization goal.
- If the statement has an OPTION clause containing a setting for the optimization_goal option, then the statement is optimized using this setting.
- Else, the optimizer uses the current setting of the option optimization_goal option.

Even if the optimization goal is First-row, the optimizer may be unable to find a plan that can quickly return the first row. For example, statements requiring materialization due to the presence of DISTINCT, GROUP BY, or ORDER BY clauses, and for which a relevant index does not exist to provide the necessary order, are optimized with the All-rows goal.

If the option is set to All-rows (the default), the query is optimized to choose an access plan with the minimal estimated total retrieval time. Setting optimization_goal to All-rows may be appropriate for applications that intend to process the entire result set, such as PowerBuilder DataWindow applications.

1.8.3.2.5 Tip: Update Column Statistics

To continually improve optimizer performance, the database server automatically updates column statistics during the processing of any SELECT, INSERT, UPDATE, or DELETE statement.

Column statistics are stored permanently in the database in the ISYSCOLSTAT system table. It does so by monitoring the number of rows that satisfy any predicate that references a table or column, comparing that number to the number of rows estimated, and then, if necessary, updating existing statistics.

With more accurate column statistics available to it, the optimizer can compute better estimates and improve the performance of subsequent queries.

You can set whether to update column statistics using database options. The update_statistics database option controls whether to update column statistics during execution of queries, while the collect_statistics_on_dml_updates database option controls whether to update the statistics during the execution of data-altering DML statements such as LOAD, INSERT, DELETE, and UPDATE.

If you suspect that performance is suffering because your statistics inaccurately reflect the current column values, execute the CREATE STATISTICS or DROP STATISTICS statements. CREATE STATISTICS deletes old statistics and creates new ones, while DROP STATISTICS only deletes old statistics.

When you execute the CREATE INDEX statement, statistics are automatically created for the index.

When you execute the LOAD TABLE statement, statistics are automatically created for the table.

In this section:

1.8.3.2.5.1 How the Statistics Governor Maintains Statistics

The statistics governor automatically evaluates the health and usefulness of each statistic in the database and performs required maintenance so that the statistics are self-monitored and self-healing.

Statistics maintenance is performed in the background and does not create a significant load on database server performance.

The statistics governor performs the following tasks:

- Records statistics usage and estimation errors from query feedback
- Fixes or remakes statistics that have low accuracy
- Stops automatic maintenance for statistics that cannot be maintained efficiently
- Creates potentially useful statistics
- Drops unused statistics

The update_statistics option controls whether the specified connection can send query feedback to the statistics governor. If this option is set to Off, the statistics governor does not receive query feedback from the specified connection. However, the statistics governor can still receive query feedback from other connections and perform maintenance operations on statistics.

The statistics governor decides when to fix or create a statistics based on its health and usage. A statistic can be fixed or created either by gathering statistics during query execution, or by a separate process called the statistics cleaner. You can disable the statistics cleaner by using the StatisticsCleaner option for the sa_server_option system procedure without disabling the statistics governor, but when the statistics cleaner is turned off, statistics are only created or fixed when a query is run.

To reduce server workload, the statistics governor stops maintenance on statistics that are hard to fix or never used. Statistics that have been fixed numerous times within a short period of time and still return poor estimates are dropped and are not maintained for 30 days. Dropped statistics are recreated after 30 days, and regular maintenance is resumed. You can disable this feature using the DropBadStatistics option for the sa_server_option system procedure. Statistics that have not been used in the last 90 days are also dropped. To disable this feature, use the DropUnusedStatistics option for the sa_server_option system procedure. You can resume maintenance on a statistic at any time by using the CREATE STATISTICS, DROP STATISTICS, or ALTER STATISTICS statements.

Statistics are only monitored for tables that are loaded into memory, and these statistics are flushed every 30 minutes. During flushing, the health and usage of the statistics are checked, and the statistics governor

performs maintenance on the statistics. The state information about a statistic (such as health, usage, and information about when to update or drop a statistic) does not persist between sessions. The state information is reset when the database server shuts down.

1.8.3.2.6 Tip: Use Indexes Effectively

The database server can evaluate search conditions with the aid of indexes.

Using indexes speeds optimizer access to data and reduces the amount of information read and processed from base tables. For example, if a query contains a search condition WHERE `column-name=value`, and an index exists on the column, an index scan can be used to read only those rows of the table that satisfy the search condition. Indexes also improve performance dramatically when joining tables.

When executing a query, the database server chooses how to access each table. When the database server cannot find a suitable index, it resorts to scanning the table sequentially, a process that can take a long time.

For example, suppose you search a large database for employees, and you only know their first or last name, but not both. If no index exists, the database server scans the entire table. However, if you created two indexes (one that contains the last names first, and a second that contains the first names first), the database server scans the indexes first, and can return the information faster.

Proper Selection of Indexes Can Make a Large Performance Difference

Although indexes let the database server locate information very efficiently, exercise some caution when adding them. Each index creates extra work every time you insert, delete, or update a row because the database server must also update all affected indexes.

Consider adding an index when it allows the database server to access data more efficiently. In particular, add an index when it eliminates unnecessarily accessing a large table sequentially. If, however, you need better performance when you add rows to a table, and finding information quickly is not an issue, use as few indexes as possible.

Use the Index Consultant to guide you through the selection of an effective set of indexes for your database.

Query Optimization

Whenever possible, the optimizer attempts index-only retrieval to satisfy a query. With index-only retrieval, the database server uses only the data in the indexes to satisfy the query, and does not need to access rows in the table. The optimizer automatically chooses to use the indexes it determines will lead to the best performance. However, you can also use index hints in your query to specify the indexes you want the optimizer to use. If any of the specified indexes cannot be used, an error is returned. Index hinting can result in poor performance and should only be attempted by experienced users. Use the Index Consultant to determine whether additional indexes are recommended for your database.

Clustered Indexes

Using clustered indexes helps store rows in a table in approximately the same order as they appear in the index.

1.8.3.2.7 Tip: Improve Index Performance

You can improve your index performance by reorganizing composite indexes or increasing the page size.

These measures improve index selectivity and index fan-out.

1.8.3.2.8 Tip: Optimize for Mixed or OLAP Workload

Use the optimization_workload option to specify optimization preference.

You can control whether query processing should be optimized towards databases where updates, deletes, or inserts are commonly executed concurrently with queries (mixed workload) or whether the main form of update activity in the database is batch-style updates that are rarely executed concurrently with query execution.

1.8.3.2.9 Tip: Use Strategic Sorting of Query Results

Use strategic sorting of query results to reduce the amount of unnecessary sorting of data.

Unless you need the data returned in a predictable order, do not specify an ORDER BY clause in SELECT statements. Sorting requires extra time and resources to process the query.

1.8.3.2.10 Tip: Specify the Correct Cursor Type

Specify the correct cursor type to improve performance.

For example, if a cursor is read-only, then declaring it as read-only allows for faster optimization and execution, since there is less material to build, such as check constraints, and so on. If the cursor is updatable, some query rewrites can be skipped. Also, if a query is updatable, then depending on the execution plan chosen by the optimizer, the database server must use a keyset driven approach. Keep in mind that keyset cursors are more expensive.

1.8.3.2.11 Tip: Supply Explicit Selectivity Estimates Sparingly

Although user defined estimates can improve performance, avoid supplying explicit user-defined estimates in statements that are to be used on an ongoing basis.

Should the data change, the explicit estimate may become inaccurate and may force the optimizer to select poor plans.

If you have used selectivity estimates that are inaccurate as a workaround to performance problems where the software-selected access plan was poor, you can set user_estimates to Off to ignore the values.

Occasionally, statistics may become inaccurate. This condition is most likely to arise when only a few queries have been executed since a large amount of data was added, updated, or deleted. Inaccurate or unavailable statistics can impede performance. If the database server is taking too long to update the statistics, try executing CREATE STATISTICS or DROP STATISTICS to refresh them.

The database server also updates some statistics when executing LOAD TABLE statements, during query execution, and when performing update DML statements.

In unusual circumstances, however, these measures may prove ineffective. If you know that a condition has a success rate that differs from the optimizer's estimate, you can explicitly supply a user estimate in the search condition.

1.8.3.2.12 Tip: Use Materialized Views to Improve Query Performance

Consider using materialized views for frequently executed, expensive queries, such as those involving intensive aggregation and join operations.

Materialized views provide a queryable structure in which to store aggregated, joined data. Materialized views are designed to improve performance in environments where the database is large, and where frequent queries result in repetitive aggregation and join operations on large amounts of data. For example, materialized views are ideal for use with data warehousing applications.

The optimizer maintains a list of materialized views to consider as candidates for partially or fully satisfying a submitted query when optimizing. If the optimizer finds a candidate materialized view that can satisfy all or part of the query, it includes the view in the recommendations it makes for the enumeration phase of optimization, where the best plan is determined based on cost. The process used by the optimizer to match materialized views to queries is called **view matching**. Before a materialized view can be considered by the optimizer, the view must satisfy certain conditions. If the optimizer determines that materialized view usage is allowed, then each candidate materialized view is examined. Unless a materialized view is explicitly referenced by the query, there is no guarantee that the optimizer uses it. You can, however, make sure that the conditions are met for the view to be considered.

In this section:

Materialized Views and View Matching [page 1453]
The optimizer uses a view matching algorithm to determine whether materialized views can be used to satisfy a query.

Retrieving the List of Materialized View Candidates (SQL) [page 1454]

Retrieve a list of materialized views that are candidates to be considered by the optimizer.

Determining Which Materialized Views Were Considered by the Optimizer [page 1455]
Determine which materialized view the optimizer used to satisfy a query.

1.8.3.2.12.1 Materialized Views and View Matching

The optimizer uses a view matching algorithm to determine whether materialized views can be used to satisfy a query.

The determination involves a query evaluation step, and a materialized view evaluation step.

Query Evaluation Step

During query evaluation, the view matching algorithm examines the query. If any of the following conditions are true, materialized views are **not** used to process the query.

- All the tables referenced by the query are updatable.
  The optimizer does not consider materialized views for a SELECT statement that is inherently updatable, or is explicitly declared in an updatable cursor. This situation can occur when using Interactive SQL, which uses updatable cursors by default for SELECT statements.
- The statement is a simple DML statement that uses optimizer bypass and is optimized heuristically. However, you can force cost-based optimization of any SELECT statement using the FORCE OPTIMIZATION option of the OPTION clause.
- For queries contained inside stored procedures and user-defined functions, the query's execution plan has been cached. The database server may cache the execution plans for these queries so that they can be reused. For this class of queries, the query execution plan is cached after execution. The next time the query is executed, the plan is retrieved and all the phases up to the execution phase are skipped.

Materialized View Evaluation Step

The optimizer includes a materialized view in the set of materialized views to be examined by the view matching algorithm if the view definition:

- contains only one query block
- contains only one FROM clause
- does not contain any of the following constructs or specifications:
  - GROUPING SETS
  - CUBE
  - ROLLUP
  - subquery
  - derived table
  - UNION

- EXCEPT
- INTERSECT
- materialized views
- DISTINCT
- TOP
- FIRST
- self-join
- recursive join
- FULL OUTER JOIN

- (optionally) contains a GROUP BY clause, and a HAVING clause, provided the HAVING clause does not contain subselects or subqueries.

In addition to meeting the view definition criteria:

- the materialized view must be enabled for use by the database server
- the materialized view must be enabled for use in optimization
- the materialized view must be initialized (populated with data)
- values for some critical options used to create the materialized views must match the options for the connection executing the query
- last refresh of the materialized view cannot have exceeded the staleness threshold set for the materialized_view_optimization database option

If the materialized view meets the above criteria, and it is found to satisfy all or part of the query, the view matching algorithm includes the materialized view in its recommendations for the enumeration phase of optimization, when the best plan is found based on cost. However, this does not mean that the materialized view will ultimately be used in the final execution plan. For example, materialized views that appear suitable for computing the result of a query may still not be used if another access plan, which doesn't use the materialized view, is estimated to be less expensive.

1.8.3.2.12.2 Retrieving the List of Materialized View Candidates (SQL)

Retrieve a list of materialized views that are candidates to be considered by the optimizer.

Procedure

1. Execute the following statement:

   ```
   SELECT * FROM sa_materialized_view_info( ) WHERE AvailForOptimization='Y';
   ```

   The list returned is specific to the requesting connection, since the optimizer takes into account option settings when generating the list. A materialized view is not considered a candidate if there is a mismatch between the options specified for the connection and the options that were in place when the materialized view was created.

2. To obtain a list of all materialized views that are not considered candidates for the connection because of a mismatch in option settings, execute the following from the connection that will execute the query:

```
SELECT * FROM sa_materialized_view_info( ) WHERE AvailForOptimization='O';
```

Results

The list of candidate materialized views is displayed.

1.8.3.2.12.3 Determining Which Materialized Views Were Considered by the Optimizer

Determine which materialized view the optimizer used to satisfy a query.

Context

> **i** Note
>
> When snapshot isolation is in use, the optimizer does not consider materialized views that were refreshed after the start of the snapshot for the current transaction.

Procedure

1. In Interactive SQL, connect to the database.
2. Click *Tools* > *Plan Viewer* (or press Shift+F5).
3. Type the query in the *SQL* pane.
4. Select a *Statistics level*, a *Cursor type*, and an *Update status*.
5. Click *Get Plan*.
6. Look on the *Details* and *Advanced Details* panes to see which materialized views, if any, were used to satisfy the query.

Results

The materialized views that were used to satisfy the query are displayed.

1.8.3.2.13 Tip: Use the WITH EXPRESS CHECK Option When Validating Tables

The WITH EXPRESS CHECK option with the VALIDATE TABLE statement, or the -fx option with the Validation utility (dbvalid) can significantly increase the speed at which your tables validate

Consider this tip if you find that validating large databases with a small cache takes a long time.

1.8.3.2.14 Tip: Use Work Tables in Query Processing (Use All-rows Optimization Goal)

Work tables are materialized temporary result sets that are created during the execution of a query.

Work tables are used when the cost is less than the alternative strategies. Generally, the time to fetch the first few rows is higher when a work table is used, but the cost of retrieving all rows may be substantially lower if a work table can be used. Because of this difference, the database server chooses different strategies based on the optimization_goal setting. The default is All-rows. When it is set to All-rows, the database server uses work tables when they reduce the total execution cost of a query. When it is set to first-row, the database server tries to avoid work tables.

Work tables are used in the following cases:

- When a query has an ORDER BY, GROUP BY, or DISTINCT clause, and the database server does not use an index for sorting the rows. If a suitable index exists and the optimization_goal setting is First-row, the database server avoids using a work table. However, when optimization_goal is set to All-rows, it may be more expensive to fetch all the rows of a query using an index than it is to build a work table and sort the rows. the database server chooses the less expensive strategy if the optimization goal is set to All-rows. For GROUP BY and DISTINCT, the hash-based algorithms use work tables, but are generally more efficient when fetching all the rows out of a query.
- When a hash join algorithm is chosen. In this case, work tables are used to store interim results (if the input doesn't fit into memory) and a work table is used to store the results of the join.
- When a cursor is opened with sensitive values. In this case, a work table is created to hold the row identifiers and primary keys of the base tables. This work table is filled in as rows are fetched from the query in the forward direction. However, if you fetch the last row from the cursor, the entire table is filled in.
- When a cursor is opened with insensitive semantics. In this case, a work table is populated with the results of the query when the query is opened.
- When a multiple-row UPDATE is being performed and the column being updated appears in the WHERE clause of the update or in an index being used for the update
- When a multiple-row UPDATE or DELETE has a subquery in the WHERE clause that references the table being modified
- When performing an INSERT from a SELECT statement and the SELECT statement references the insert table
- When performing a multiple row INSERT, UPDATE, or DELETE, and a corresponding trigger is defined on the table that may fire during the operation

In these cases, the records affected by the operation go into the work table. In certain circumstances, such as keyset-driven cursors, a temporary index is built on the work table. The operation of extracting the required records into a work table can take a significant amount of time before the query results appear. Creating

indexes that can be used to do the sorting in the first case, above, improves the time to retrieve the first few rows. However, the total time to fetch all rows may be lower if work tables are used, since these permit query algorithms based on hashing and merge sort. These algorithms use sequential I/O, which is faster than the random I/O used with an index scan.

The optimizer analyzes each query to determine whether a work table would give the best performance. No user action is required to take advantage of these optimizations.

Notes

The INSERT, UPDATE, and DELETE cases above are usually not a performance problem since they are usually one-time operations. However, if problems occur, you may be able to rephrase the statement to avoid the conflict and avoid building a work table. This is not always possible.

1.8.3.2.15 Tip: Build Efficient SQL Queries

To improve query processing performance, consider building more efficient queries.

These tips reflect optimizations that the optimizer might choose during query processing to rewrite the query more efficiently. By building these efficiencies into the query, the optimizer will likely have less work to do.

Tip	Before and after	Explanation
Eliminate unnecessary DISTINCT conditions	Before: `SELECT DISTINCT p.ID, p.Quantity FROM Products p;` After: `SELECT p.ID, p.Quantity FROM Products p;`	The DISTINCT keyword in the first statement is unnecessary because the Products table contains the primary key p.ID, which is part of the result set.

Tip	Before and after	Explanation
Eliminate unnecessary DISTINCT conditions	Before: `SELECT DISTINCT *` `FROM SalesOrders o JOIN` `Customers c` `ON o.CustomerID =` `c.ID` `WHERE c.State = 'NY';` After: `SELECT *` `FROM SalesOrders o JOIN` `Customers c` `ON o.CustomerID =` `c.ID` `WHERE c.State = 'NY';`	The first query contains the primary keys of both tables, so each row in the result must be distinct.
Un-nest subqueries	Before: `SELECT s.*` `FROM SalesOrderItems s` `WHERE EXISTS ( SELECT *` `FROM Products p` `WHERE s.ProductID =` `p.ID` `AND p.ID = 300` `AND p.Quantity > 20);` After: `SELECT s.*` `FROM Products p JOIN` `SalesOrderItems s` `ON p.ID = s.ProductID` `WHERE p.ID = 300 AND` `p.Quantity > 20;`	Rewriting nested queries as joins often leads to more efficient execution and more effective optimization. In general, subquery un-nesting is always done for correlated subqueries with, at most, one table in the FROM clause, which are used in ANY, ALL, and EXISTS predicates. A uncorrelated subquery, or a subquery with more than one table in the FROM clause, is flattened if it can be decided, based on the query semantics, that the subquery returns at most one row. In this example, the subquery can match at most one row for each row in the outer block. Because it can match at most one row, it can be converted to an inner join.

Tip	Before and after	Explanation
Un-nest subqueries	Before: `SELECT p.*` `FROM Products p` `WHERE EXISTS` `( SELECT *` `FROM` `SalesOrderItems s` `WHERE s.ProductID` `= p.ID` `AND s.ID = 2001);` After: `SELECT DISTINCT p.*` `FROM Products p JOIN` `SalesOrderItems s` `ON p.ID = s.ProductID` `WHERE s.ID = 2001;`	The Before query contains a conjunctive EXISTS predicate in the subquery, which can match more than one row. It can be converted to an inner join, with a DISTINCT in the SELECT list.
Un-nest subqueries	Before: `SELECT *` `FROM Products p` `WHERE p.ID =` `( SELECT s.ProductID` `FROM` `SalesOrderItems s` `WHERE s.ID = 2001` `AND s.LineID =` `1 );` After: `SELECT p.*` `FROM Products p,` `SalesOrderItems s` `WHERE p.ID = s.ProductID` `AND s.ID = 2001` `AND s.LineID = 1;`	Eliminate subqueries in comparisons when the subquery matches at most one row for each row in the outer block.

Tip	Before and after	Explanation
Consider using an IN predicate when querying an indexed column	Before: `SELECT * FROM SalesOrders WHERE SalesRepresentative = 902 OR SalesRepresentative = 195;` After: `SELECT * FROM SalesOrders WHERE SalesRepresentative IN ( 195, 902 );`	In the rewritten form, the IN-list predicate can be treated as a sargable predicate and exploited for indexed retrieval. Also, the optimizer can sort the IN-list to match the sort sequence of the index, leading to more efficient retrieval. The IN-list must contain only constants, or values that are constant during one execution of the query block, such as outer references.
Eliminate unnecessary joins	Before: `SELECT s.ID, s.LineID, p.ID FROM SalesOrderItems s KEY JOIN Products p FOR READ ONLY;` After: `SELECT s.ID, s.LineID, s.ProductID FROM SalesOrderItems s WHERE s.ProductID IS NOT NULL FOR READ ONLY;`	Consider eliminating joins when: • The join is a primary key to foreign key join, and only primary key columns from the primary table are referenced in the query. In this case, the primary key table is eliminated if it is not updatable. • The join is a primary key to primary key join between two instances of the same table. In this case, one of the tables is eliminated if it is not updatable. • The join is an outer join and the null-supplying table expression returns at most one row for each row of the preserved side of the outer join, and no expression produced by the null-supplying table expression is needed in the rest of the query beyond the outer join. In this case, the join is a primary key to foreign key join so the primary key table, Products, can be eliminated. That is, the second query is semantically equivalent to the first because any row from the SalesOrderItems table that has a NULL foreign key to Products does not appear in the result.

Tip	Before and after	Explanation
Eliminate unnecessary joins	Before: `SELECT s.ID, s.LineID` `FROM SalesOrderItems s` `LEFT OUTER JOIN` `Products p` `ON p.ID = s.ProductID` `WHERE s.Quantity > 5` `FOR READ ONLY;` After: `SELECT s.ID, s.LineID` `FROM SalesOrderItems s` `WHERE s.Quantity > 5` `FOR READ ONLY;`	In the first query, the OUTER JOIN can be eliminated because the null-supplying table expression cannot produce more than one row for any row of the preserved side and none of the columns from Products is used above the LEFT OUTER JOIN.
Eliminate unnecessary case translation	Before: `SELECT *` `FROM Customers` `WHERE UPPER(Surname) =` `'SMITH';` After: `SELECT *` `FROM Customers` `WHERE Surname = 'SMITH';`	On a case insensitive database, the first query can be rewritten so that the optimizer can consider using an index on Customers.Surname. By default, the database server performs case-insensitive string comparisons unless explicit text conversion instructions are given(use of UPPER, UCASE, LOWER, LCASE). Eliminating unnecessary case translations allows the predicates to be turned into sargable predicates, which can be used for index retrieval of the corresponding table.

Tip	Before and after	Explanation
Consider inlining functions	Before: `CREATE FUNCTION F1( arg1 INT, arg2 INT ) RETURNS INT BEGIN RETURN arg1 * arg2 END; SELECT F1( e.EmployeeID, 2.5 ) FROM Employees e;` After: `SELECT CAST( e.EmployeeID AS INT ) * CAST( 2.5 AS INT ) FROM Employees e;`	You can inline user-defined functions if they take one of the following forms: • contains a single RETURN statement • declares a single variable, assigns the variable, and returns a single value • declares a single variable, selects into that variable, and returns a single value This tip is not applicable to temporary functions, recursive functions, or functions with a NOT DETERMINISTIC clause. This tip is also not applicable if the function is called with a subquery as an argument, or when it is called from inside a temporary procedure.
Consider inlining simple stored procedures	Before: `CREATE PROCEDURE Test1( arg1 INT ) BEGIN SELECT * FROM Employees WHERE EmployeeID=arg1 END; SELECT * FROM Test1( 200 );` After: `SELECT * FROM ( SELECT * FROM Employees WHERE EmployeeID=CAST( 200 AS INT ) ) AS Test1;`	You can inline a stored procedure that is defined only as a single SELECT statement when calling it in the FROM clause of a query. When a procedure is inlined, it is rewritten as a derived table. This tip does not apply to procedures that use default arguments, that contain anything other than a single SELECT statement in the body.

1.8.3.2.16 Tip: Monitor Query Performance

Several tools for testing the performance of queries are provided.

Use the Profiler to monitor queries in stored procedures, functions, triggers, and events. The sa_get_request_profile and sa_get_request_times system procedures also measure query execution times.

The following table lists additional tools that are stored in subdirectories under *%SQLANYSAMP17%* `\SQLAnywhere`. Complete documentation about each tool can be found in a `readme.txt` file that is located in the same folder as the tool.

Tool	Function	Location
fetchtst	Determines the time required for a result set to be retrieved.	*%SQLANYSAMP17%* `\SQLAnywhere \PerformanceFetch`
odbcfet	Determines the time required for a result set to be retrieved. This tool is similar to fetchtst, but with less functionality.	*%SQLANYSAMP17%* `\SQLAnywhere \PerformanceFetch`
instest	Determines the time required for rows to be inserted into a table.	*%SQLANYSAMP17%* `\SQLAnywhere \PerformanceInsert`
trantest	Measures the load that can be handled by a given database server configuration given a database design and a set of transactions.	*%SQLANYSAMP17%* `\SQLAnywhere \PerformanceTransaction`

1.8.3.2.17 Tip: Reduce Expensive User-defined Functions

Reducing expensive user-defined functions in queries where they have to be executed many times can improve performance.

1.8.3.2.18 Tip: Turn Off Autocommit Mode

Instead of running in autocommit mode, consider grouping your SQL statements so each group performs one logical task.

If your application runs in **autocommit mode**, then the database server treats each of your statements as a separate transaction. In effect, it is equivalent to appending a COMMIT statement to the end of each of your statements.

If you disable autocommit, you must execute an explicit commit after each logical group of SQL statements. Also, be aware that if logical transactions are large, blocking and deadlock can happen.

If you are not using a transaction log file, the cost of using autocommit mode is high. Every statement forces a checkpoint, an operation that can involve writing numerous pages of information to disk.

Each application interface has its own way of setting autocommit behavior. For the Open Client, ODBC, and JDBC interfaces, Autocommit is the default behavior.

1.8.3.2.19 Tip: Use In-memory Mode

If your application can tolerate the loss of committed transactions after the most recent checkpoint, then your application may benefit from using in-memory mode.

This mode is useful in applications where increased performance is desirable, and you are running on a system with a large amount of available memory, typically enough to hold all the database files within the cache.

You can choose between two different in-memory modes. In never-write mode, committed transactions are not written to the database file on disk. When you specify never-write mode, multiple concurrent LOAD TABLE statements can be active on the same or different tables. All changes are lost if the database is shut down or the connection is lost. In checkpoint-only mode, the database server does not use a transaction log, and you cannot recover to the most recent committed transaction. However, because the checkpoint log is enabled, the database can be recovered to the most recent checkpoint.

In-memory mode requires a separate license.

1.8.3.2.20 Tip: Check the Lagtime Mirror Option Setting (Mirrored Systems Only)

The lagtime mirror option can impact performance on mirrored systems.

If performance is slow in a mirroring system, then it could be related to the lagtime mirror option. This setting can act as a throttle to slow the rate of transactions on the primary server when the mirror server is lagging too far behind. If you suspect that the lagtime mirror option is impacting performance, then consider setting the option to a larger value.

1.8.3.2.21 Tip: Always Use a Transaction Log

Using a transaction log can provide data protection, and can improve the performance of the database server.

When operating without a transaction log, SQL Anywhere performs a checkpoint at the end of every transaction, and that checkpoint may consume considerable resources.

When operating with a transaction log, the database server only writes notes detailing the changes as they occur. It can choose to write the new database pages all at once, at the most efficient time. **Checkpoints** make sure information enters the database file, and that it is consistent and up to date.

> ⚠ Caution
>
> The database file and the transaction log file must be located on the same physical computer as the database server or accessed via a SAN or iSCSI configuration. Database files and transaction log files located on a remote network directory can lead to poor performance, data corruption, and server instability.

1.8.3.2.22 Tip: Use Delayed Commits

Delayed commits determine the rate of response a database provides to a COMMIT statement.

When the rate of committed changes to a database is high, the rate of transaction log writes can be the single largest factor in determining overall database performance. If you are trying to improve transaction log performance, you can set the delayed_commits option to On. When set to On, the database server replies to a COMMIT statement immediately instead of waiting until the transaction log entry for the COMMIT has been written to disk. When set to Off, the application must wait until the COMMIT is written to disk. Turning on the delayed_commits option results in fewer transaction log writes by avoiding multiple re-writes of partially filled log pages, and you can set the option per connection or for all connections. When the delayed_commits option is turned on, there is a risk that committed operations may be lost if the server goes down before the transaction log pages are flushed to disk.

1.8.3.2.23 Tip: Collect Statistics on Small Tables

By default, the database server creates statistics for tables with more than five rows. To create statistics for a table with less than five rows, use the CREATE STATISTICS statement.

Statistics gathered while processing one statement are available when searching for efficient ways to execute subsequent statements.

1.8.3.2.24 Tip: Reduce Fragmentation

Fragmentation occurs as you make changes to your database, and performance can suffer if your files, tables, or indexes are excessively fragmented.

Reducing fragmentation becomes more important as your database increases in size.

If you are noticing a significant decrease in performance, consider:

- rebuilding your database to reduce table and/or index fragmentation, especially if you have performed extensive delete/update/insert activity on multiple tables
- putting the database on a disk partition by itself to reduce file fragmentation
- running one of the available Windows utilities periodically to reduce file fragmentation
- reorganizing your tables to reduce database fragmentation
- using the REORGANIZE TABLE statement to defragment rows in a table, or to compress indexes which may have become sparse due to DELETEs. Reorganizing tables can reduce the total number of pages used to store a table and its indexes, and it may reduce the number of levels in an index tree as well.

In this section:

Reduce File Fragmentation [page 1466]
Reducing disk fragmentation becomes more important as the size of your database increases.

Reduce Table Fragmentation [page 1466]

Table fragmentation occurs when rows are not stored contiguously, or when rows are split between multiple pages.

Methods to Reduce Table Fragmentation [page 1467]
Table fragmentation is controlled through several methods.

Viewing an Object's Fragmentation Details [page 1468]
View fragmentation information for an object from the *Fragmentation* tab in SQL Central.

Reorganizing Base Tables and Indexes [page 1469]
Reorganize base tables and indexes from the *Fragmentation* tab in SQL Central.

Reduce Index Fragmentation and Skew [page 1470]
Indexes are designed to speed up searches on particular columns, but they can become fragmented (less dense) and skewed (unbalanced) if many delete operations are performed on the indexed table.

1.8.3.2.24.1 Reduce File Fragmentation

Reducing disk fragmentation becomes more important as the size of your database increases.

The database server determines the number of file fragments in each dbspace when you start a database on Windows. The database server displays the following performance warning in the database server messages window when the number of fragments is greater than one: `Database file "mydatabase.db" consists of nnn disk fragments.`

You can also obtain the number of database file fragments using the DBFileFragments database property.

To eliminate file fragmentation problems, put the database on a disk partition by itself and then periodically run one of the available Windows disk defragmentation utilities.

1.8.3.2.24.2 Reduce Table Fragmentation

Table fragmentation occurs when rows are not stored contiguously, or when rows are split between multiple pages.

These rows require additional page access and this reduces the performance of the database server.

The effect that fragmentation has on performance varies. A table might be highly fragmented, but if it fits in memory, and the way it is accessed allows the pages to be cached, then the impact may be minimal. However, a fragmented table may cause much more I/O to be done and can significantly reduce performance if split rows are accessed frequently and the cost of extra I/Os is not reduced by caching.

While reorganizing tables and rebuilding a database can reduce fragmentation, doing so too frequently or not frequently enough, can also impact performance. Experiment using the following tools and methods to determine an acceptable level of fragmentation for your tables.

If you reduce fragmentation and performance is still poor, another issue may be to blame, such as inaccurate statistics.

Determine the Degree of Table Fragmentation

Checking the table fragmentation just once is not helpful in determining whether to defragment to improve performance. Instead, rebuild your database and check the table fragmentation to establish baseline results. Then, continue to check the table fragmentation periodically over an extended length of time, looking for correlation between the change in fragmentation to changes in performance measures. This method helps you determine the rate at which tables become fragmented to the degree that performance is impacted, and so determine the optimal frequency at which to defragment tables.

To obtain information about the degree of fragmentation of your database tables, use one of the following methods:

- Call the sa_table_fragmentation system procedure. For example:

  ```
  CALL sa_table_fragmentation( );
  ```

- The *Fragmentation* tab in the SQL Anywhere plug-in. The *Fragmentation* tab provides a graphical representation of the results from running sa_table_fragmentation system procedure on base tables.

1.8.3.2.24.3 Methods to Reduce Table Fragmentation

Table fragmentation is controlled through several methods.

Use PCTFREE

The database server reserves extra room on each page to allow rows to grow slightly. When an update to a row causes it to grow beyond the original space allocated for it, the row is split and the initial row location contains a pointer to another page where the entire row is stored. For example, filling empty rows with UPDATE statements or inserting new columns into a table can lead to severe row splitting. As more rows are stored on separate pages, more time is required to access the additional pages.

You can reduce the amount of fragmentation in your tables by specifying the percentage of space in a table page that should be reserved for future updates. This PCTFREE specification can be set with CREATE TABLE, ALTER TABLE, DECLARE LOCAL TEMPORARY TABLE, or LOAD TABLE.

Reorganize Tables

You can defragment specific tables using the REORGANIZE TABLE statement or clicking *Reorganize* on the *Fragmentation* tab in SQL Central.

Rebuild the Database

Rebuilding the database defragments all tables, including system tables, *provided the rebuild is performed as a two-step process*, that is, data is unloaded and stored to disk, and then reloaded. Rebuilding in this manner also has the benefit of rearranging the table rows so they appear in the order specified by the clustered index and primary keys. One-step rebuilds (for example, using the -ar, -an, or -ac options), do not reduce table fragmentation.

In this section:

The Fragmentation Tab (SQL Anywhere Plug-in) [page 1468]

You can use the *Fragmentation* tab to view the fragmentation of base tables and indexes on those tables, and to reorganize tables and indexes.

1.8.3.2.24.3.1 The *Fragmentation* Tab (SQL Anywhere Plug-in)

You can use the *Fragmentation* tab to view the fragmentation of base tables and indexes on those tables, and to reorganize tables and indexes.

Zooming Within a Dbspace Map

By default, when a dbspace map is opened in the bottom pane of the *Fragmentation* tab, the zoom level is set to *Fit To Window*. You can zoom in by clicking the dbspace map and you can zoom out by pressing Shift while clicking. When you click or press Shift while clicking within the dbspace map, the clicked page is centered in the map after the zoom level change.

You can also use the toolbar buttons to zoom to the following levels:

Toolbar button	Definition
1: 1	*1 page: 1 pixel*
64KB: 1	*1 64KB block: 1 pixel* (one database read)
Fit To Window	Uses all available space in the window

1.8.3.2.24.4 Viewing an Object's Fragmentation Details

View fragmentation information for an object from the *Fragmentation* tab in SQL Central.

Prerequisites

You must have the CHECKPOINT system privilege.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, select the database. In the right pane, click the *Fragmentation* tab.
3. Select an object from the top pane. The fragmentation information appears in a dbspace map in the bottom pane:
   - When you select a base table, the table, its extension pages, and applicable index pages appear in the dbspace map in the bottom pane.
   - When you select an index, its index pages appear in the dbspace map in the bottom pane.
4. Click *Checkpoint & Refresh* to perform a checkpoint and see the most recent fragmentation information.

5. View the page indexes:
   - In the dbspace map in the bottom pane, hover your cursor over a colored-vertical bar to see the first and last page indexes at that position.
   - In the dbspace map, press and hold the Ctrl key while hovering the mouse over a colored-vertical bar to see all the page indexes at that position.

Results

Fragmentation details for the specified object appear in the dbspace map in the bottom pane.

1.8.3.2.24.5 Reorganizing Base Tables and Indexes

Reorganize base tables and indexes from the *Fragmentation* tab in SQL Central.

Prerequisites

You must be the owner of the object or have the REORGANIZE ANY OBJECT system privilege.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, select the database. In the right pane, click the *Fragmentation* tab.
3. Select an object from the top pane. The fragmentation information appears in a dbspace map in the bottom pane.
4. Choose one of the following methods to reorganize the object:
   - Click *Reorganize* to execute a REORGANIZE TABLE statement on the selected object.
   - Copy an object from the top pane into an Interactive SQL *SQL Statements* pane. A REORGANIZE TABLE statement for the object appears in the *SQL Statements* pane. Execute the statement. This method is useful when you want to reorganize the objects at a later time or when you want to continue using SQL Central while reorganizing the objects.

Results

The specified table or index is reorganized.

1.8.3.2.24.6 Reduce Index Fragmentation and Skew

Indexes are designed to speed up searches on particular columns, but they can become fragmented (less dense) and skewed (unbalanced) if many delete operations are performed on the indexed table.

Index density reflects the average fullness of the index pages. Index skew reflects the typical deviation from the average density. The amount of skew is important to the optimizer when making selectivity estimates.

To determine whether your database contains indexes that contain unacceptable levels of fragmentation or skew, use the Profiler.

You can also use the sa_index_density system procedure to review levels of index fragmentation and skew. For example, the following statement calls the sa_index_density system procedure to examine indexes on the Customers table.

```
CALL sa_index_density( 'Customers', 'GROUPO' );
```

TableName	TableId	IndexName	IndexID	IndexType	LeafPages	Density	Skew
Customers	718	Customer-sKey	0	PKEY	1	0.12768554 6875	1
Customers	718	IX_cus-tomer_name	1	NUI	1	0.78955078 125	1

The database server creates indexes on primary keys automatically. These indexes have an IndexID of 0 in the results for the sa_index_density system procedure.

When the number of leaf pages is low, you do not need to be concerned about density and skew values. Density and skew values become important only when the number of leaf pages is high. When the number of leaf pages is high, a low density value can indicate fragmentation, and a high skew value can indicate that indexes are not well balanced. Both of these can be factors in poor performance. Executing a REORGANIZE TABLE statement addresses both of these issues.

You can also use the *Fragmentation* tab in the SQL Anywhere plug-in to review levels of index fragmentation on indexes associated with base tables.

1.8.3.2.25 Tip: Normalize Your Table Structure

Database tables may contain multiple copies of the same information (for example, a column that is repeated in several tables), and your table may need to be normalized.

Normalization reduces duplication in a relational database. For example, suppose your company employees work at several different offices. To normalize the database, consider placing information about the offices (such as its address and main telephone numbers) in a separate table, rather than duplicating all this information for every employee.

If the amount of duplicate information is small, you may find it better to duplicate the information and maintain its integrity using triggers or other constraints.

1.8.3.2.26 Tip: Minimize Cascading Referential Actions

Cascading referential actions are costly because they cause updates to multiple tables for every transaction. This can affect performance.

For example, if the foreign key from Employees to Departments was defined with ON UPDATE CASCADE, then updating a department ID would automatically update the Employees table. While cascading referential actions are convenient, sometimes it might be more efficient to implement them in application logic instead.

1.8.3.2.27 Tip: Declare Constraints

Declaring primary key and foreign key relationships and their constraints can improve performance.

This is especially true when joins take place because the cost model is able to do a better job of estimation.

Not declaring the relationship can save time on index maintenance. Undeclared primary key-foreign key relationships exist between tables when there is an implied relationship between the values of columns in different tables.

1.8.3.2.28 Tip: Place Different Files on Different Devices

You can improve database performance by putting different database files on different physical devices or drives.

For example, while one disk drive is busy swapping database pages to and from the cache, another drive can be writing to the transaction log file. To gain these benefits, the drives must be independent. A single disk partitioned into smaller logical drives is unlikely to yield benefits.

Disk drives operate much more slowly than modern processors or RAM. Often, simply waiting for the disk to read or write pages is the reason that a database server is slow.

The database server uses four types of files: the **database file**, the **transaction log file**, the **transaction log mirror**, and the **temporary file**. These files should exist on separate drives.

Placing the database file and the transaction log file on physically separate drives is recommended to protect against media failure.

Placing the transaction log mirror file and the temporary file on physically separate drives can help the database server run faster. The database server writes more efficiently to the transaction log and transaction log mirror files when they exist on separate drives. When the database server needs to use the temporary file, the overall database performance is heavily dependent on the speed of the drive containing the temporary file. Because many operations that use the temporary file also require retrieving information from the database, placing the temporary file on a separate drive allows the operations to take place simultaneously.

A database can be held in up to 13 separate files (the main file and 12 dbspaces), which can be located on separate drives. Place tables into separate dbspaces so that common join operations read information from different dbspaces.

When you create all tables or indexes in a location other than the system dbspace, the system dbspace is only used for the checkpoint log and system tables. This configuration is useful to put the checkpoint log on a

separate drive from the rest of your database objects for performance reasons. To create base tables in another dbspace, change all the CREATE TABLE statements to use the IN DBSPACE clause to specify the alternative dbspace, or change the setting of the default_dbspace option before creating any tables. Temporary tables can only be created in the TEMPORARY dbspace.

A similar strategy involves placing the temporary and database files on a RAID device or a stripe set. Although such devices act as a logical drive, they dramatically improve performance by distributing files over many physical drives and accessing the information using multiple heads.

You can specify the -fc option when starting the database server to implement a callback function when the database server encounters a file system full condition.

1.8.3.2.29 Tip: Rebuild Your Database

Rebuilding your database can improve performance.

Rebuilding improves the organization of data.

1.8.3.2.30 Tip: Use Keys to Improve Query Performance

Primary keys and foreign keys can also improve database performance.

Example

The following example illustrates how primary keys can make queries execute more quickly.

```
SELECT *
FROM Employees
WHERE EmployeeID = 390;
```

The simplest way for the database server to execute this query would be to look at all 75 rows in the Employees table and check the employee ID number in each row to see if it is 390. This does not take very long since there are only 75 employees, but for tables with many thousands of entries a sequential search can take a long time.

The referential integrity constraints embodied by each primary or foreign key are enforced by the database server through the help of an index, implicitly created with each primary or foreign key declaration. The EmployeeID column is the primary key for the Employees table. The corresponding primary key index permits the retrieval of employee number 390 quickly. This quick search takes almost the same amount of time whether there are 100 rows or 1000000 rows in the Employees table.

Separate indexes are created automatically for primary and foreign keys. This arrangement allows the database server to perform many operations more efficiently.

1.8.3.2.31 Tip: Reduce Primary Key Width

Reducing the number of columns in your primary keys can improve performance.

Wide primary keys are composed of two or more columns. The more columns contained in your primary key, the more demand there is on the database server.

1.8.3.2.32 Tip: Reduce Table Widths

If you have wide tables, and find performance slow consider further normalizing your tables to reduce the number of columns.

If that is not possible, a larger database page size may be helpful, especially if most tables are wide

Tables where the combined columns (or the size of an individual row) exceeds the database page size and must be split across two or more database pages are referred to as wide table. The more pages a row takes up, the longer the database server takes to read each row.

1.8.3.2.33 Tip: Review the Order of Columns in Tables

The order of the columns in a table affects performance.

Columns in a row are accessed sequentially in the order of their creation. For example, to access columns at the end of a row, the database server traverses the columns that appear earlier in the row. Order your columns so that narrow and/or frequently accessed columns are placed before seldom accessed and/or wider columns in the table.

Wide columns are columns greater than 15 bytes in size, or LONG data types (for example, LONG VARCHAR), or columns defined as XML. Primary key columns are always stored at the beginning of row.

1.8.3.2.34 Tip: Replace Expensive Triggers

Evaluate the use of triggers to see if some of the triggers could be replaced by features available in the database server.

For instance, triggers to update columns with the latest update time and user information can be replaced with the corresponding special values in the database server. As well, using the default settings on existing triggers can also improve performance.

1.8.3.2.35 Tip: Use an Appropriate Page Size

The database page size can affect the performance of your database.

There are advantages and disadvantages to both large and small page sizes.

The database server attempts to fill pages as much as possible. Empty space accumulates only when new objects are too large to fit empty space on existing pages. So, adjusting the page size may not significantly affect the overall size of your database.

It is strongly recommended that you test performance (and other behavior aspects) when choosing a page size. Then, choose the smallest page size that gives satisfactory results. It is important to pick the correct and reasonable page size if more than one database is started on the same server.

Smaller pages hold less information and may use space less efficiently, particularly if you insert rows that are slightly more than half a page in size. However, small page sizes allow the database server to run with fewer resources because more pages can be stored in a cache of the same size. Small pages are useful if your database runs on a small computer with limited memory. They can also help when your database is used primarily for the retrieval of small pieces of information from random locations.

A larger page size helps the database server read databases more efficiently. Large page sizes tend to benefit large databases, and queries that perform sequential table scans. Often, the physical design of disks permits them to retrieve fewer large blocks more efficiently than many small ones. Other benefits of large page sizes include improving the fan-out of your indexes, thereby reducing the number of index levels, and allowing tables to include more columns. If you choose a larger page size, consider increasing the size of the cache because fewer large pages can fit into a cache of the same size. If your cache cannot hold enough pages, performance suffers as the database server begins swapping frequently used pages to disk.

Larger page sizes have additional memory requirements. As well, extremely large page sizes (16 KB or 32 KB) are not recommended for most applications unless you can be sure that a large database server cache is always available.

The database server's memory usage is proportional to the number of databases loaded, and the page size of the databases. It is strongly recommended that you do performance testing (and testing in general) when choosing a page size. Then choose the smallest page size (>= 4 KB) that gives satisfactory results. It is important to pick the correct (and reasonable) page size if a large number of databases are going to be started on the same server.

You cannot change the page size of an existing database. Instead you must create a new database and use the -p option of dbinit to specify the page size. For example, the following command creates a database with 4 KB pages.

```
dbinit -dba DBA,passwd -p 4096 new.db
```

You can also use the CREATE DATABASE statement with a PAGE SIZE clause to create a database with the new page size.

For each table, the database server creates a bitmap that reflects the position of each table page in the entire dbspace file. The database server uses the bitmap to read large blocks (64 KB) of table pages, instead of single pages at a time. This efficiency, also known as **group reads**, reduces the total number of I/O operations to disk, and improves performance. Users cannot control the database server's criteria for bitmap creation or usage.

Page Size and Indexes

Page size also affects indexes. Each index lookup requires one page read for each of the levels of the index plus one page read for the table page, and a single query can require several thousand index lookups. Page size can significantly affect fan-out, in turn affecting the depth of index required for a table. A large fan-out often means that fewer index levels are required, which can improve searches considerably. For large databases that have tables with a significant numbers of rows, 8 KB pages may be warranted for the best performance.

Scattered Reads

If you are working with Windows, a minimum page size of 4 KB allows the database server to read a large contiguous region of database pages on disk directly into the appropriate place in cache, bypassing the 64 KB buffer entirely. This feature can significantly improve performance.

> **i** Note
>
> Scattered reads are not used for files on remote computers, or for files specified using a UNC name such as `\\mycomputer\myshare\mydb.db`.

1.8.3.2.36 Tip: Use AUTOINCREMENT to Create Primary Keys

Using the AUTOINCREMENT feature to generate primary key values is faster than other methods because the value is generated by the database server.

You can use the AUTOINCREMENT default for any column in which you want to maintain unique values.

1.8.3.2.37 Tip: Use Appropriate Data Types

You can improve performance by using the appropriate data type for your data.

Data types store information about specific sets of data, including ranges of values, the operations that can be performed on those values, and how the values are stored in memory. For instance, avoid assigning a data type of CHAR to values that only contain numeric data. And whenever possible, choose efficient data types over the more expensive numeric and string types.

1.8.3.2.38 Tip: Use Bulk Operations Methods

If you load large amounts of information into your database, you can benefit from the special tools provided for these tasks.

If you are loading large files, it is more efficient to create indexes on the table after the data is loaded.

1.8.3.2.39 Tip: Use Resource Governors

Resource governors can be used to improve performance.

Resource governors are a set of database options you can use to control resources. These options can be set using the SET OPTION statement.

Building a set of users and roles allows you to manage privileges on a database. Another aspect of database security and management is to limit the resources an individual user can use.

For example, you can prevent a single connection from taking too much of the available memory or CPU resources, so you can avoid having a connection slow down other users of the database.

Resources That Can Be Managed

You can use the following options to manage resources:

max_cursor_count

Limits the number of cursors for a connection.

max_statement_count

Limits the number of prepared statements for a connection.

priority

Sets the priority level at which requests from a connection are executed.

max_priority

Controls the maximum priority level for connections.

1.8.3.2.40 Tip: Reduce Requests Between Client and Server

Use the LazyClose and PrefetchOnOpen network connection parameters to reduce the number of requests between the client and server.

This can improve performance when your network exhibits poor latency, or your application sends many cursor open and close requests.

1.8.3.2.41 Tip: Use Compression Carefully

Enabling compression for one connection or all connections, and adjusting the minimum size limit at which packets are compressed can offer significant improvements to performance.

Compression of Packets for Connections

To determine if enabling compression is beneficial, conduct a performance analysis on your network and using your application before using communication compression in a production environment.

Enabling compression increases the quantity of information stored in data packets, thereby reducing the number of packets required to transmit a particular set of data. By reducing the number of packets, the data can be transmitted more quickly.

Specifying the compression threshold allows you to choose the minimum size of data packets that you want compressed. The optimal value for the compression threshold may be affected by a variety of factors, including the type and speed of network you are using.

Database Compression

The use of file-level or disk-level compression for database and log files is not recommended since the compression layer may significantly increase the cost of IO operations.

1.8.3.2.42 Tip: Change Packet Size to Improve Performance

In some cases, increasing the packet size can improve performance.

In some cases, performance can decrease. Increasing the packet size also increases the amount of memory used by both the client and the database server. In a production environment, conduct a performance analysis on your network using your applications before you make any adjustments.

Increasing the packet size may improve database request response times, especially for requests that transfer a large amount of data between a client and a database server. For example, an application that has many large (more than 64 KB) result sets or that transfers many large (more than 64 KB) BLOBs over very fast local networks may have a measurable performance increase by increasing the packet size from the default.

You can set the packet size by using the -p dbeng17/ dbsrv17 server option, or by setting the CommBufferSize (CBSIZE) connection parameter in your connection string.

1.8.3.3 Optimize Indexes to Improve Performance

Obtain recommendations on the best set of indexes for your database by running the index analyzer tool, the Index Consultant.

The selection of a proper set of indexes can improve database performance.

The Index Consultant guides you through the process of selecting indexes for a single query or for a set of database requests (called a workload). It generates candidate indexes and determines their effect on performance. To explore the effect of different candidate indexes, the Index Consultant repeatedly re-optimizes the queries under different sets of indexes. It does not execute the queries.

To see which indexes lead to improved execution plans, the Index Consultant estimates query execution costs using those indexes. It also evaluates multiple column indexes, single-column indexes, and investigates the impact of clustered or unclustered indexes.

In this section:

1.8.3.3.1 Obtaining Index Recommendations for Queries (Profiler)

Obtain index recommendations for one or more queries (SELECT, UPDATE, or DELETE statements) by using the Profiler.

Prerequisites

You must have the SYS_RUN_PROFILER_ROLE system role.

Procedure

1. Start Profiler and connect to the database.

1. Run the following command to start Interactive SQL and connect to the database:.

   ```
   dbisql -c "START=dbeng17 -x TCPIP;UID=DBA;PWD=sql;DBF=C:\Users\Public
   \Documents\SQL Anywhere 17\Samples\demo.db"
   ```

2. In Interactive SQL, click *Tools* > *Profiler*.
3. When prompted, accept the default profiling options and click *OK*.

Profiling information from your database begins to appear in the Profiler.

2. In Interactive SQL, execute a SELECT, UPDATE, or DELETE statement.
3. In Profiler, chose one of the following options:

Option	Description
Generate index recommendations for all queries that have run since profiling started.	1. Click *Tools* > *Suggest index for workload* and follow the instructions in the wizard.
Generate index recommendations for a specific query.	1. Open the *Statements* tab and select a SELECT, UPDATE, or DELETE statement. 2. Click *Tools* > *Suggest index for statement* and follow the instructions in the wizard.

4. When prompted, select the type of index recommendations that you want to receive:

   Recommend clustered indexes

   When this option is selected, the Index Consultant analyzes the effect of clustered and unclustered indexes.

   Properly selected clustered indexes can provide significant performance improvements over unclustered indexes for some workloads, but you must reorganize the table (using the REORGANIZE TABLE statement) for them to be effective. In addition, the analysis takes longer if the effects of clustered indexes are considered.

   Keep existing secondary indexes

   The Index Consultant can perform its analysis by either maintaining the existing set of secondary indexes in the database, or by ignoring the existing secondary indexes. A secondary index is an index that is not a unique constraint or a primary or foreign key. Indexes that are present to enforce referential integrity constraints are always considered when selecting access plans.

5. Follow the instructions in the wizard.

   Proposed changes are displayed, and a SQL script to implement any changes is provided.

   Analyze any reported errors encountered while running the Index Consultant by looking at the sql_code and log_message columns in the "DBO"."ix_consultant_log" table.

6. Review the proposed changes against your knowledge of the database to ensure that the proposals will improve performance. You can edit the SQL script to customize it for your database.

   You can rename the proposed index names generated during the analysis.

7. Run the SQL script to implement the changes.

1.8.3.3.2 Obtaining Index Recommendations for Queries (Interactive SQL)

Access Index Consultant recommendations for a query in Interactive SQL.

Prerequisites

You must have the DIAGNOSTICS system role or all of the following system privileges:

- SELECT ANY TABLE
- INSERT ANY TABLE
- DELETE ANY TABLE
- UPDATE ANY TABLE

Procedure

1. In Interactive SQL, connect to the database.

   ```
   dbisql -c "START=dbeng17 -x TCPIP;UID=DBA;PWD=sql;DBF=C:\Users\Public
   \Documents\SQL Anywhere 17\Samples\demo.db"
   ```

   .
2. In the *SQL Statements* pane, execute a SELECT, UPDATE, or DELETE statement.
3. Click *Tools* > *Index Consultant* and follow the instructions in the wizard.

Results

Index recommendations appear in the *Summary* pane of the *Index Consultant Wizard*.

Next Steps

Review the recommendations to ensure that they satisfy needs of your database before running the script to implement them.

1.8.3.3.3 Improving Performance by Executing a List of CREATE INDEX or a List of LOAD TABLE Statements Concurrently

Use the BEGIN PARALLEL WORK statement to execute a list of CREATE INDEX and LOAD TABLE statements in parallel.

Prerequisites

You must have the privileges necessary to execute the statements in the list.

Context

Statements listed inside a BEGIN PARALLEL WORK statement execute in parallel. The number of statements that can execute at the same time is limited by the number of logical processors on the computer that the database server runs on, and it is further limited by the max_parallel_statements option, the -gta and -gtc database server options, and the ProcessorAffinity option of the sa_server_option system procedure.

When unloading a database by using the Unload utility (dbunload), specify the -bp option to group CREATE INDEX and LOAD TABLE statements inside BEGIN PARALLEL WORK statements in the `reload.sql` file.

Procedure

Choose one of the following options:

Option	Actions
Execute a list of CREATE INDEX statements in parallel.	Execute a BEGIN PARALLEL WORK statement with a list of CREATE INDEX statements. For example: `BEGIN PARALLEL WORK` `  CREATE INDEX L_SHIPDATE_IDX` `  ON LINEITEM(l_shipdate);` `  CREATE INDEX O_ORDERDATE_IDX` `  ON ORDERS(o_orderdate);` `  ...` `END PARALLEL WORK;`
Execute a list of LOAD TABLE statements in parallel.	Execute a BEGIN PARALLEL WORK statement with a list of LOAD TABLE statements. For example: `BEGIN PARALLEL WORK`

Option	Actions
	`` LOAD TABLE dba.Part FROM 'D:\\data\\part.tbl' FORMAT 'ASCII' QUOTES OFF ESCAPES ON STRIP OFF HEXADECIMAL OFF DELIMITED BY '|' ORDER OFF; LOAD TABLE dba.Supplier FROM 'D:\\data\\supplier.tbl' FORMAT 'ASCII' QUOTES OFF ESCAPES ON STRIP OFF HEXADECIMAL OFF DELIMITED BY '|' ORDER OFF; ... END PARALLEL WORK; ```

Results

The BEGIN PARALLEL WORK statement along with the statements listed inside of it, executes.

Related Information

BEGIN PARALLEL WORK Statement
CREATE INDEX Statement
LOAD TABLE Statement
max_parallel_statements Option [page 763]
Unload Utility (dbunload) [page 1233]

1.8.4 Diagnostics

You can use a number of tools to diagnose problems in a database and database server.

In this section:

SQL Anywhere Profiler [page 1483]
The Profiler is a development and diagnostic tool that logs the activities that occur in your database in real time and analyzes the information for performance issues.

Other Diagnostic Tools and Techniques [page 1500]
In addition to Profiler, a variety of other diagnostic tools and techniques are available to help you analyze and monitor the current performance of your database.

1.8.4.1 SQL Anywhere Profiler

The Profiler is a development and diagnostic tool that logs the activities that occur in your database in real time and analyzes the information for performance issues.

Use the Profiler to help identify such issues as:

- Deadlocks and blocked connections
- Long-running and expensive queries, as well as repeatedly run statements
- Expensive hidden procedures, for example, triggers, events, and nested stored procedure calls
- Potential problem areas within the body of a procedure

Use the Profiler to collect information about all database activity (comprehensive profiling) or to collect a subset of executing SQL statements (targeted profiling). Both options collect statistics about the database server load and automatically analyze your workload to identify or rule out issues that could be affecting your application's performance. Always start your performance investigation by reviewing this workload analysis.

Comprehensive Versus Targeted Profiling

Comprehensive profiling collects the most information about your workload. It collects all activity that occurs in your database, including information about connections, SQL statements, events, console messages, web server messages, and internal database server operations. Comprehensive profiling provides you with the most visibility into the activities occurring in your database. In contrast, **targeted profiling** collects only information about the database server load and the SQL statements that fulfill your specified criteria.

Because so much information is collected during a comprehensive profiling session, the performance of the database being monitored is affected. In contrast, targeted profiling collects information about only the SQL statements that fulfill specific criteria. Because targeted profiling collects less information, performance is minimally affected. You can run a targeted profiling session for an extended period of time on a database in a production environment.

Comprehensive profiling is the only option that provides features for profiling stored SQL objects and reviewing blocked objects and connections. Its analysis of the workload is more extensive because it has additional types of information to consider. However, targeted profiling allows you to create the collection criteria for the SQL statements to collect.

Which Profiling Option Should I Choose?

Targeted profiling is often a good choice when you are concerned about the performance of your database, especially if you are profiling a database while it is running in a production environment. However, when the information collected by targeted profiling is not sufficient, use comprehensive profiling and limit the length of the profiling sessions. Comprehensive profiling is the more powerful option because it collects information about all activity occurring in the database and it provides more features such as the ability to profile stored SQL objects.

Choose targeted profiling if you are mainly interested in information about your database server load and in the Profiler's analysis of your application. Targeted profiling is a great choice when you know the characteristics of

the SQL statements that are likely to be affecting performance. It can also be a good choice when you have a complex application that generates a lot of database activity, because it reduces the amount of information that you need to filter to identify the information that is of interest to you.

Comprehensive profiling is useful when you are investigating a performance problem that has a number of potential causes. Use comprehensive profiling when you are developing your application to validate your designs.

In this section:

Related Information

Diagnostic Tracing (Deprecated) [page 1510]
SQL Anywhere Cockpit [page 1419]
Profiler Utility (dbprof) [page 1199]

1.8.4.1.1 Running a Comprehensive Profiling Session (Profiler)

Collect information about all activity occurring in your database, investigate how the objects in your database interact with each other, and review the Profiler's analysis of the issues that are affecting the performance of your database.

Prerequisites

You must have the SYS_RUN_PROFILER_ROLE system role to use the Profiler.

Context

During a comprehensive profiling session, the Profiler collects information about connections, SQL statements, events, console messages, web server messages, and internal server operations.

This option collects the text plan for each executed statement. You can only view graphical plans during the profiling session; text plans are available after you disconnect from the Profiler and are included when you save your session to a `.sqlap` file.

Procedure

1. Start the Profiler by clicking *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Anywhere Profiler*.
2. Connect to the database.

   For example, in the *Connect* window, choose *Connect with a connection string*, and then specify the following connection parameters:

   ```
   UID=DBA;PWD=sql;"DSN=SQL Anywhere 17 Demo"
   ```
3. When prompted, choose *Comprehensive*.
4. Run your application.
5. Open the *Analysis* tab, click *Workload Summary*, and then review the analysis of your workload.
6. Save your profiling data by clicking *File* > *Save As*.

Next Steps

Disconnect from the Profiler by clicking *File* > *Disconnect*.

1.8.4.1.2 Running a Targeted Profiling Session (Profiler)

Limit the information collected to database server performance statistics and SQL statements that fulfill your specified criteria.

Prerequisites

You must have the SYS_RUN_PROFILER_ROLE system role to use the Profiler.

Context

Because less information is collected during a targeted profiling session, there is less information for you to filter inside the Profiler, and the performance of the profiled database is minimally affected. You can run a targeted profiling session for an extended period of time, which makes it useful for databases running in a production environment. The Profiler also analyzes and reports on your database server's workload.

Graphical plans that are collected as part of a targeted profiling session are included when you save your results to a `.sqlap` file.

Procedure

1. Start and connect to the Profiler by clicking *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Anywhere Profiler*.

   For example, in the *Connect* window, choose *Connect with a connection string*, and then specify the following connection parameters:

   ```
   UID=DBA;PWD=sql;"DSN=SQL Anywhere 17 Demo"
   ```
2. When prompted, choose *Targeted*, and adjust the criteria as required so that the Profiler only collects information about statements that fulfill the specified criteria.

   By default, graphical plans are collected, but you can choose to collect text plans instead.
3. Run your application.
4. Open the *Statements* tab in the Profiler, and view the statements that fulfilled your criteria.
5. In the top pane of the *Statements* tab, click a row associated with a statement, that you want to investigate.

   Detailed information for each execution of the selected statement appears in the bottom table.
6. Double click a row in the bottom pane to view the details about a specific execution of the statement.

   A statement can run slowly because the database server chose a poor execution plan. If a graphical plan was created for the statement, it appears on the *Plan* tab.
7. Save your results by clicking *File* > *Save As*.

8. Stop profiling by clicking *File* > *Disconnect*.

Example

When prompted to choose the profiling option, choose *Targeted*, and then specify **2** for the *Statement is active for at least 10 seconds* field.

In Interactive SQL, connect to the database and execute the following long-running statement:

```
SELECT COUNT(*) FROM customers t1, customers t2, customers t3, customers t4;
```

In the Profiler, open the *Statements* tab. The executed statement is listed in the table.

In this section:

1.8.4.1.2.1 Criteria for Specifying the SQL Statements to Collect During a Targeted Profiling Session (Profiler)

Use the following criteria to define the characteristics of the SQL statements that you want the Profiler to collect during a targeted profiling session.

When you choose to run a targeted profiling session, the default is to collect SQL statements that run for more than 10 seconds. Adjust this threshold and specify other characteristics for the Profiler to use by using the criteria listed in the table below. The Profiler only collects SQL statements that fulfill the characteristics that you specify.

In the Profiler, the statement properties for a SQL statement collected during a targeted profiling session include properties that have the same names and definitions as the criteria listed in this table.

Criteria	Description
active_time	Time in seconds that the database server spent executing the statement.
actual_rows	Number of rows returned by the statement.
approx_cpu_time	Approximate time in seconds that the statement used the CPU.
blocked_io_count	Number of times that the statement blocked waiting for I/O.

Criteria	Description
blocked_lock_count	Number of times that the statement blocked waiting for a database lock.
blocked_lock_time	Time in seconds that the statement blocked waiting for a database lock. For example 10 or 10.55.
conn_id	Connection ID that executed the statement.
dbn	Name of database.
elapsed_time	Wall clock time in seconds between the first fetch and most recent fetch of the cursor.
estimated_io	Estimated number of I/O that are needed to execute the statement.
estimated_rows	Estimated number of rows to be returned by the statement. A negative number indicates that the optimizer doesn't have an accurate estimate of the maximum number of rows to be returned.
estimated_time	Estimated time, in seconds, that is needed to execute the statement.
procedure_creator	Name of the user that created the procedure, function, trigger, or event.
procedure_name	Value of the CurrentProcedure connection property.
line_number	Line number (1..n) within the procedure.
plan	Plan for the statement with the format associated with the event type.
plan_signature	Number that characterizes the execution plan. This number depends on the join strategy.
sql_string	The SQL string associated with the statement. The text is generated by unparsing and does not exactly match the original.
stmt_handle	Statement handle that you use to link to other trace events.
traceback	Stack trace of the procedures/user functions.
user	Value of the UserID connection property.

1.8.4.1.3 Comparing the Results of Different Executions of Stored Procedures, Functions, Events, and Triggers (Profiler)

Profile your stored SQL objects, such as procedures, functions, events, and triggers to identify the objects or statements that could be affecting the performance of your system. Once you have identified a potential solution, test it, and then compare your results.

Prerequisites

You must have the SYS_RUN_PROFILER_ROLE system role to use the Profiler.

Context

To view the actual SQL statements that comprise a stored SQL object, open the *Profiling* tab. For each stored SQL object that executed, this tab shows each line of code as well as, statistics about each line's execution times.

Procedure

1. Start and connect to the Profiler by clicking *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Anywhere Profiler*.

   For example, in the *Connect* window, choose the action *Connect with a connection string*, and then specify the following connection parameters:

   ```
   UID=DBA;PWD=sql;DSN=SQL Anywhere 17 Demo
   ```
2. When prompted, choose *Comprehensive*.
3. Open the *Profiling* tab.

   The top pane lists in a table the SQL objects that have executed since profiling started.
4. In your application, execute the stored SQL object which you want to profile.
5. In the top pane, click the row associated with the stored SQL object that you executed.

   Statistics specific to that SQL object appear in the in lower pane, including the time it took for each line of the stored SQL object to execute.

   If the stored SQL object executed multiple times, then the table also contains the time differences between the baseline execution of the SQL object and any subsequent executions. By default, the first execution of the stored SQL object is used as the baseline until you specify a different execution. Baseline values are cleared when you reconnect to the Profiler.
6. Review the contents of the lower pane and create a new baseline by clicking *Set as Baseline*. Often lines with long execution times compared to other lines in the code should be analyzed to see whether there is a more efficient way to achieve the same functionality.
7. Make a change to the code in your application, and then rerun the stored SQL object in your application.
8. Observe the performance changes on the *Profiling* tab in the Profiler.
9. Continue changing your application and re-running it until you are satisfied with your changes.
10. Save your profiling results by clicking *File* > *Save as*.

1.8.4.1.4 Sending SAP Support a Snapshot of Your Database Server'S Diagnostics (Profiler)

When directed by SAP Support, capture diagnostic information about your database server's internal performance statistics, save this information to a file, and then send it to SAP Support.

Prerequisites

You must have the SYS_RUN_PROFILER_ROLE system role to use the Profiler.

Context

The information that you collect includes the actual SQL code of the most expensive statements that have executed in your database since the database server started. It also includes any graphical plans that are associated with these statements.

> ⚠ Caution
>
> Review the snapshot's contents before sending it to Technical Support. Depending upon the SQL statements that were executed, you could reveal your database schema, intellectual property, and possibly sensitive data. For example, if one of your expensive statements is an INSERT statement that uses the VALUES clause to insert sensitive information, then this sensitive information appears in the snapshot and is saved to `.sqlap` file.

Procedure

1. Start and connect to the Profiler by clicking *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Anywhere Profiler*.

   For example, in the *Connect* window, choose the action *Connect with a connection string*, and then specify the following connection parameters:

   ```
   UID=DBA;PWD=sql;"DSN=SQL Anywhere 17 Demo"
   ```

2. When prompted, select the *Support* option, and then click *OK*.

   This option collects graphical plans that are included when you save this session to a `.sqlap` file.

3. Optional. In the Profiler, open the *Analysis* tab, and then click *Statement Performance Summary* to view the snapshot.

   In the top pane, a table lists the most expensive statements that have executed since the database server started. Click a row to view the performance summary information for a statement, including any graphical plans that were used.

4. Click *File* > *Save as* and save the file to your computer.
5. Send the `.sqlap` file to SAP Support as directed.

1.8.4.1.5 Troubleshooting: Detect When Hardware Resources Affect Performance by Using the Workload Summary and Server Load Graph (Profiler)

Run a profiling session, and then review the results of the Workload Summary to learn whether hardware resources are a limiting factor for performance.

As larger and larger workloads are placed on a database, performance is impacted by CPU cycles, disk I/O bandwidth, and even network I/O bandwidth. An inefficient application or database server could also contribute to performance problems. If you cannot detect any inefficiencies, then you may need to add additional hardware resources.

Adding resources may not resolve scalability problems or improve computer performance. For example, if a database server is using all of its allotted CPUs, it may indicate that more CPU resources are required. However, doubling the number of CPUs available to the database server may not double the amount of work the database server can perform.

When investigating a potential hardware issue, run a profiling session, and then review the results of the Workload Summary by opening the *Analysis* tab and clicking *Workload Summary*. The Workload Summary automatically scans your workload and identifies times when your server appears to be CPU, disk, network, or database worker-bound. You can then drill down into each interval to see what operations the server was processing.

The Workload Summary can identify when the following conditions are symptoms of performance problems:

CPU usage is high

If for an extended period of time, the CPU percentage is greater than 90% while the disk I/O and network I/O are low, then your system could be CPU bound. Check the *Profiling* and *Analysis* tabs for long-running queries that could be revised to run faster.

If the CPU usage and the number of active requests is high while the disk I/O and network I/O usages are normal, then consider upgrading the hardware or setting up a mirroring or scale-out solution to spread out the load.

Disk I/O usage is high

If for an extended period of time, your disk I/O usage is unusually high while your CPU usage and network I/O usage are low, then your system could be disk I/O bound. Often, adding more hardware to your system, such as faster hard disks or using a RAID array to speed up disk I/O, can improve your performance. Or, the problem could be resolved by adjusting your application. For example, disk I/O performance issues can be the result of a cache size that is too small or indexes that are improperly configured. Click *Tools* > *Suggest indexes for workload* to produce a set of recommended indexes that could increase the throughput of your application.

Network usage is high

If for an extended period of time your network I/O usage is unusually high while your CPU usage and Disk I/O usage are low, then you could be network bound. You could investigate your network setup or consider

adjusting your application. For example, there could be is an issue with your network cards, or you could be sending an excessive amount of data over the network.

The number of active requests is high

A high number of active requests means that you have a large number of connections that are all trying to do something in the database.

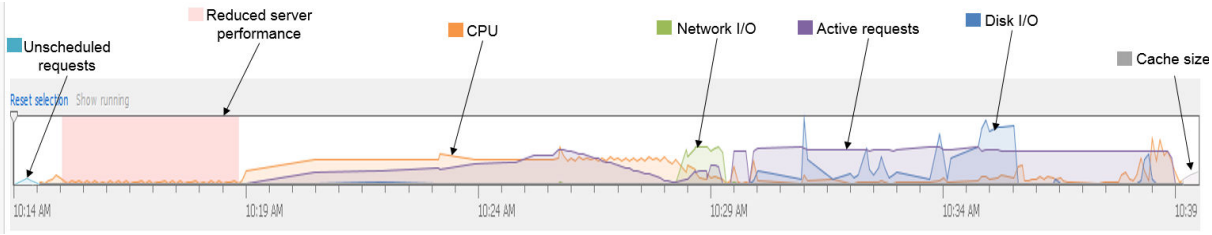
If for an extended period of time, the number of active requests is high and the CPU is low, then there are a number of things that you can investigate, such as:

- Blocked and deadlocked connections. Check the *Blocking* tab for issues related to blocked or deadlocked connections.
- Database errors.
- Long-running statements that could be optimized for specific queries.

Server Load Graph

You can also inspect the raw CPU, disk, network, and server request data yourself in the *Server Load* panel. The *Server Load* graph tracks the percentage of CPU, disk I/O, and network I/O usage, as well as the number of active requests, unscheduled requests, and the cache size. Depending upon your system, high values for these statistics can indicate normal or problematic conditions.

Below is an example of a *Server Load* graph.



When interpreting the *Server Load* graph, consider the metrics together to get an overall picture of what is going on in your system. Few conclusions can be obtained by examining the metrics in isolation. Choose what values to display and how to configure the graph by using the *Presets* dropdown menu in the *Server Load* panel.

In the line graphs for these metrics, the vertical height of the lines reflects only the usage amounts. The Profiler does not know the capacity of your CPUs, disks, or network. Use these line graphs to identify usage trends, and then hover your mouse over the graph to see the absolute values.

Sometimes performance issues occur because the database server is running an internal process, such as a backup, which can lock the catalog. Pink lines or areas in the *Server Load* graph indicate periods of reduced database server performance that correspond to internal processes, such as issuing checkpoints, running backups, running the cleaner, resizing the cache, reorganizing a table, and changing the multiprogramming level. If your performance issue disappears with the pink line, then an internal process was likely creating the problem.

If your performance issue is caused by an internal process and the degradation is unacceptable, then there are a number of things that you can investigate, from rescheduling the processes to occur during off-peak times, to database mirroring and scale-out solutions.

Related Information

Tips for Improving Performance [page 1436]
Performance Profiling Tutorials (Profiler) [page 1493]
Tip: Use the Cache to Improve Performance [page 1440]
Tip: Use Indexes Effectively [page 1450]
Tip: Build Efficient SQL Queries [page 1457]
Tip: Use an Appropriate Page Size [page 1474]
Optimize Indexes to Improve Performance [page 1478]

1.8.4.1.6 Troubleshooting: Application Logic Problems (Profiler)

If you have errors in your application code or in procedures, triggers, functions, or events, then it is useful to examine all statements executed by the database server that relate to the incorrect code.

For applications that dynamically generate SQL, use the Profiler to examine the actual SQL that is sent to the database server to help you detect errors in how the SQL is built by the application. Such errors may cause queries to fail to be executed, or a query may return different results than it was intended to return. For example, during development, your application may occasionally report that it encountered a SQL syntax error, but your application may not be instrumented to report the SQL text of the query that failed. If you start the Profiler and run the application, then you can search for statements that returned syntax (or other) errors, and see the exact text that was generated by your application.

To analyze database objects, such as procedures and triggers, you can also use the debugger in SQL Central. However, there may be times when it is more effective to use the Profiler to trace all the statements executed by a given procedure. For example, a given stored procedure may be returning an incorrect result once out of every 1000 invocations, but you may not understand under what conditions it fails. Rather than step through the procedure code 1000 times in the debugger, you could start the Profiler, and run your application. Then, you could examine the set of statements that the database server executed, locate the set of statements that correspond to the incorrect execution of the procedure, and determine either why the procedure failed, or the conditions under which it behaves unexpectedly. If you know under what conditions the procedure behaves unexpectedly, then you can set a breakpoint in the procedure and investigate further with the debugger.

1.8.4.1.7 Performance Profiling Tutorials (Profiler)

Use these tutorials to learn how to analyze common performance problems, including deadlocks, slow statements, index fragmentation, and slow procedures.

In this section:

Tutorial: Collecting Profiling Information (Profiler) [page 1494]
Learn how to collect and view profiling information about your database using the Profiler.

Tutorial: Diagnosing Blocked Connections and Deadlocks (Profiler) [page 1496]

Learn how to use the Profiler to identify blocked connections and deadlocks that occur in a database.

Tutorial: Profiling Procedures (Profiler) [page 1498]
Compare the execution times of different implementations of a stored procedure to determine which version performs better.

1.8.4.1.7.1 Tutorial: Collecting Profiling Information (Profiler)

Learn how to collect and view profiling information about your database using the Profiler.

Prerequisites

You must have the SYS_RUN_PROFILER_ROLE system role to use the Profiler.

Context

The Profiler displays information about currently executing statements.

Procedure

1. Start Interactive SQL and connect to the sample database as the default DBA user.
   a. In Interactive SQL, click *SQL* > *Connect*.
   b. In the *Action* dropdown list, choose *Connect with a connection string*, and specify the following connection parameters:

      ```
      UID=DBA;PWD=sql;"DSN=SQL Anywhere 17 Demo"
      ```
2. Start profiling the database by starting the Profiler and connecting to the database.
   a. Click *Tools* > *Profiler*.
   b. When prompted, choose *Comprehensive*.

      This option collects the most comprehensive information about the database server, and it allows you to investigate deadlocks and blocked connections and to profile your stored SQL.

   Diagnostic information collected from the database appears in the Profiler.
3. In the Profiler, open the *Analysis* tab and click *Workload Summary* to review an analysis of your workload that highlights areas on which to focus to improve performance.

   Always start your investigation by reviewing the workload summary. The workload summary addresses questions about whether performance is limited by factors such as:

   - blocking

- database server hardware
- server workers
- backups, checkpoints or other internal database server activity

4. Open the *Operations* tab.

   This tab lists all the operations that have run in your database since you started profiling it.

   The bar chart in the right pane displays the length of time it took to process the operation. If an operation is blocked, then the bar chart displays striped lines for the time it is blocked. To view more information about a statement or operation, double-click the operation to open its properties window.

   Some internal operations in the database server are known and expected to affect performance, such as backing up the database, checkpointing, resizing the cache, running the cleaner, reorganizing a table, and changing the multiple programming level. When these internal processes run, the background of the right pane appears red to indicate those times when you should expect reduced server performance.

5. In Interactive SQL, execute the following statement:

   ```
   SELECT COUNT( * ) FROM customers t1, customers t2;
   ```

6. In the Profiler, reduce the information displayed in the *Operations* tab to only the operations related to the statement you just ran.
   a. Click in the *Filter* box, and then click *Edit Filter Expression* from the dropdown menu.
   b. Create a filter that only shows executed statements that took longer than 5 seconds to execute.
7. Click *File* > *Save as* to save your profiling results to a file.
8. Stop the profiling session.
   a. Click *File* > *Disconnect*.
   b. When prompted, choose to stop profiling before disconnecting.

   The Profiler stops collecting information. You can still access the information that it collected during your profiling session, except for any graphical plans.

Results

You have collected profiling information.

Next Steps

Proceed to the next tutorial.

1.8.4.1.7.2 Tutorial: Diagnosing Blocked Connections and Deadlocks (Profiler)

Learn how to use the Profiler to identify blocked connections and deadlocks that occur in a database.

Prerequisites

You must have the SYS_RUN_PROFILER_ROLE system role to use the Profiler.

Context

Deadlocks occur when two or more transactions block one another. For example, Transaction A requires access to Table B, but Table B is locked by Transaction B. Transaction B requires access to Table A, but Table A is locked by Transaction A. A cyclical blocking conflict occurs.

A good indication that deadlocks are occurring is when SQLCODE -306 and -307 are returned. To resolve a deadlock, the database server automatically rolls back the last statement that created the deadlock. Performance problems occur if statements are constantly rolled back.

Procedure

1. Start Interactive SQL and connect to the sample database as the default DBA user.
   a. In Interactive SQL, click *SQL* > *Connect*.
   b. In the *Action* dropdown list, choose *Connect with a connection string*, and specify the following parameters:

```
UID=DBA;PWD=sql;"DSN=SQL Anywhere 17 Demo"
```

2. Configure the database so that it is easy to create a deadlock.
   a. Execute the following SQL statements to create two tables and two procedures that you will run later to create a deadlock.

```
CREATE OR REPLACE TABLE "DBA"."deadlock1" (
   "id" UNSIGNED BIGINT NOT NULL DEFAULT AUTOINCREMENT,
   "val" CHAR(1) );
CREATE OR REPLACE TABLE "DBA"."deadlock2" (
   "id" UNSIGNED BIGINT NOT NULL DEFAULT AUTOINCREMENT,
   "val" CHAR(1) );
INSERT INTO "deadlock1"("val") VALUES('x');
INSERT INTO "deadlock2"("val") VALUES('x');
CREATE OR REPLACE PROCEDURE "DBA"."proc_deadlock1"(  )
   BEGIN
      LOCK TABLE "DBA"."deadlock1" IN EXCLUSIVE MODE;
      WAITFOR DELAY '00:00:20.000';
      UPDATE deadlock2 SET val='y';
   END;
```

```
CREATE OR REPLACE PROCEDURE "DBA"."proc_deadlock2"(  )
   BEGIN
      LOCK TABLE "DBA"."deadlock2" IN EXCLUSIVE MODE;
      WAITFOR DELAY '00:00:20.000';
      UPDATE deadlock1 SET val='y';
   END;
COMMIT;
```

3. Start profiling the database by starting the Profiler and connecting to the database.
   a. Click *Tools* > *SQL Anywhere Profiler*.
   b. Click *OK* to accept the profile option defaults and start profiling the database.

   Diagnostic information collected from the database appears in the *Operations* tab in the Profiler.
4. In Interactive SQL, create another connection to the database.
   a. Click *Window* > *New Tab*.
   b. Connect to the sample database by specifying **DBA** for the user name and **sql** for the password.
5. Create a cyclical deadlock as follows:
   a. On the first Interactive SQL tab, execute the following SQL statement:

      ```
      CALL "DBA"."proc_deadlock1"();
      ```

   b. On the second Interactive SQL tab, execute the following SQL statement within 20 seconds of executing the SQL statement in the first Interactive SQL tab:

      ```
      CALL "DBA"."proc_deadlock2"();
      ```

      A deadlock occurs and:

      - On the second Interactive SQL tab, an error message appears.
      - On the first Interactive SQL tab, the locking indicator in the status bar displays two locks. Click the locking indicator to view details.

      The deadlock occurred because the proc_deadlock1 procedure requires access to the deadlock2 table, which is locked by the proc_deadlock2 procedure. At the same time, the proc_deadlock2 procedure requires access to the deadlock1 table, which is locked by the proc_deadlock1 procedure.
6. View the deadlock in the Profiler.
   a. On the *Operations* tab, find the executed statement labeled **CALL "DBA"."proc_deadlock1"()**.

      The bar chart in the right pane displays the length of time it took to process the statement. If the statement is blocked, then the bar chart displays striped lines for the time it is blocked. To view more information about a statement, double-click the statement.
   b. Click the *Blocking* tab and then click *Deadlocks*.

      The top table groups together statements that have been involved in deadlocks since you started profiling the database.

      Detailed information for each invocation of the selected statement appears in the bottom table.
   c. Click the executed statement labeled **CALL "DBA"."proc_deadlock1"()** to view the details in the bottom pane.
7. Determine if performance is limited by blocking by clicking the *Summary* pane in the *Analysis* tab.

   The *Summary* pane provides an analysis of your workload that highlights areas on which to focus to improve performance. If performance is limited by blocking, then the statements that block for the longest

amount of time are shown. You can configure the amount of time for which a statement must be blocked to be reported by clicking the *Configure* icon.

8. Stop profiling by clicking *File* > *Disconnect*.

Results

You created a deadlock and collected information about it.

Next Steps

Optional. Restore the sample database to its original state by deleting the stored procedures that you created in this tutorial.

1.8.4.1.7.3 Tutorial: Profiling Procedures (Profiler)

Compare the execution times of different implementations of a stored procedure to determine which version performs better.

Prerequisites

You must have the SYS_RUN_PROFILER_ROLE system role to use the Profiler

Context

Procedure profiling shows you how long it takes your procedures, user-defined functions, events, system triggers, and triggers to execute. You can view line-by-line execution times for these objects. Once you determine which objects should be fine tuned to improve performance within your database, set your saved results as a baseline. Then make incremental changes to your code and rerun it after each change. By comparing the new results to the baseline, you can verify if your changes work as intended.

Procedure

1. Start Interactive SQL and connect to the sample database as the default DBA user.
   a. In Interactive SQL, click *SQL* > *Connect*

b. In the *Action* dropdown list, choose *Connect with a connection string*, and specify the following parameters:

```
UID=DBA;PWD=sql;"DSN=SQL Anywhere 17 Demo"
```

2. Start profiling the database by starting the Profiler and connecting to the database.
   a. Click *Tools* > *SQL Anywhere Profiler*.
   b. Click *OK* to accept the profile option defaults and start profiling the database.

   Diagnostic information collected from the database begins to appear in the Profiler.
3. In Interactive SQL, execute the following SQL statement to create a stored procedure to test and then run the stored procedure.
   a. Execute the following statements to create the procedure longproc:

```
CREATE OR REPLACE PROCEDURE LONGPROC ()
BEGIN
DECLARE LOCAL TEMPORARY TABLE RET (pkey INT, fkey INT);
FOR lp AS CRSR CURSOR FOR SELECT EmployeeID AS emp_id FROM GROUPO.Employees
DO
    INSERT INTO ret
    SELECT emp_id, so."ID"
    FROM GROUPO.SalesOrders so
    WHERE so.SalesRepresentative = emp_id;
END FOR;
SELECT *
FROM ret;
END;
```

   b. Execute the following statement to call the longproc procedure:

```
CALL DBA.longproc ();
```

4. In the Profiler, click the *Profiling* tab.

   The top table of the *Profiling* tab groups together the stored SQL objects, such as procedures, functions, triggers, and events that have run in your database since profiling began.

   To find stored SQL objects that are running slowly on your system examine the *Count* and *Time* columns. These columns provide information about the execution times for object processed by the database server. Focus on those stored SQL objects that run slowly each time that they are executed, or on those objects that are executed frequently.
5. Click the row in the top pane that corresponds to the CALL longproc statement.

   The SQL code for the stored procedure appears in the bottom pane. The information displayed includes the execution times for each line in the SQL code.
6. Click *Set as baseline* to create a baseline to compare your results.

   The content in the *Profiling* tab clears.
7. Edit the longproc procedure to make it more efficient, and then test your results.
   a. In Interactive SQL, execute the following statements:

```
CREATE OR REPLACE PROCEDURE LONGPROC ()
BEGIN
SELECT e.EmployeeID, so."ID" FROM GROUPO.Employees e, GROUPO.SalesOrders so
WHERE so.SalesRepresentative = e.EmployeeID;
END;
```

b. Execute the following statement to run the new longproc procedure:

```
CALL DBA.LONGPROC ()
```

8. Review your results in the bottom pane of the *Profiling* tab in the Profiler.

   The delta columns that appear for the *Count* and *Time* columns result from comparing statistics from the baseline to the statistics captured during the most recent execution of the procedure. Specifically, they compare number of executions and duration of execution, respectively, for each line of code in the procedure.

   In this tutorial, the SELECT statement in the updated procedure has a faster time than the SELECT statement in the baseline procedure.

9. Stop profiling by clicking *File* > *Disconnect*.

Next Steps

Optional. Restore the sample database (demo.db) to its original state by deleting the procedures that you added in this tutorial.

1.8.4.2 Other Diagnostic Tools and Techniques

In addition to Profiler, a variety of other diagnostic tools and techniques are available to help you analyze and monitor the current performance of your database.

Other methods for analyzing performance data included:

Tool	Details
Diagnostic tracing	**Database tracing** Database tracing provides the ability to customize the type of performance data gathered. You can use this tool to monitor the performance of specific users or activities. **Index Consultant** This feature analyzes the indexes in the database and provides recommendations for improvement. You can access this tool through the Profiler or through Interactive SQL. **Procedure profiling** You can also use SQL statements to perform procedure profiling.

Tool	Details
Other tools	**Request logging** Request logging logs to a text file individual requests received from, and responses sent to, an application. It is most useful for determining what the database server is being asked to do by the application. Request logging is also a good starting point for performance analysis of a specific application when it is not obvious whether the database server or the client is at fault. You can use request logging to determine the specific request to the database server that might be responsible for problems. The request log provides a subset of the information that is provided by diagnostic tracing and event tracing. **Event tracing** Event tracing is recommended for production environments and provides fine-grained control over the information that is logged. You can log both user- and system-defined trace events for both the database server and your application and customize the trace events to identify performance issues. **Execution plans** This feature allows you to examine the execution plan to access information in the database related to a statement. You can view the execution plan in Interactive SQL or use SQL functions. You can retrieve an execution plan in several different formats and the plan can be saved.

In this section:

Monitoring Database Servers (Windows Performance Monitor) [page 1521]
Use the Windows Performance Monitor to view counters related to your database, server, or connection.

Identifying and Fixing Index Fragmentation (SQL) [page 1523]
Use the sa_index_density stored procedure to identify index fragmentation, and then rebuild the indexes to fix the problem.

Identifying and Fixing Table Fragmentation (SQL) [page 1524]
Determine if your database has table fragmentation, and if necessary, learn how to fix table fragmentation

1.8.4.2.1 Request Logging

Request logging logs individual requests received from, and responses sent to, an application.

It is most useful for determining what the database server is being asked to do by the application.

Request logging is also a good starting point for performance analysis of a specific application when it is not obvious whether the database server or the client is at fault. You can use request logging to determine the specific request to the database server that might be responsible for problems.

> **i Note**
>
> All the functionality and data provided by the request logging feature is also available using diagnostic tracing. Diagnostic tracing also offers additional features and data.

Logged information includes such things as timestamps, connection IDs, and request type. For queries, it also includes the isolation level, number of rows fetched, and cursor type. For INSERT, UPDATE, and DELETE statements, it also includes the number of rows affected and number of triggers fired.

> **i Note**
>
> The request log can contain statements with obfuscated sensitive information. If you store passwords or keys in plain text to the server, then that information is logged. The only case when sensitive information is not obfuscated is a statement with a parsing error.

You can use the -zr server option to turn on request logging when you start the database server. You can redirect the output to a request log file for further analysis using the -zo server option. The -zn and -zs option let you specify the number of request log files that are saved and the maximum size of request log files. Additionally, you can use the sa_server_option system procedure to set up request logging on a server that is already running.

> **i Note**
>
> These server options do not impact diagnostic tracing in SQL Central. File-based request logging is completely separate from the diagnostic tracing feature in SQL Central, which makes use of dbo-owned diagnostic tables in the database to store request log information.

The sa_get_request_times system procedure reads a request log and populates a global temporary table (SATMP_request_time) with statements from the log and their execution times. For INSERT/UPDATE/DELETE statements, the time recorded is the time when the statements were executed. For queries, the time recorded

is the total elapsed time from PREPARE to DROP (describe/open/fetch/close). Request times are not provided for cursors that are still open.

Analyze SATMP_request_time for statements that could be candidates for improvements. Statements that are inexpensive, but frequently executed, may represent performance problems.

You can use sa_get_request_profile to call sa_get_request_times and summarize SATMP_request_time into another global temporary table called SATMP_request_profile. This procedure also groups statements together and provides the number of calls, execution times, and so on.

> ⚠ Caution
>
> If the log is being analyzed using the `tracetime.pl` Perl script, the max_client_statements_cached option should be set to 0 to disable client statement caching while the request log is captured.

Example

Output to the request log can be filtered to include only requests from a specific connection or from a specific database, using the sa_server_option system procedure. This can help reduce the size of the log when monitoring a database server with many active connections or multiple databases.

To filter according to a connection:

```
CALL sa_server_option( 'RequestFilterConn' , connection-id );
```

You can obtain `connection-id` by executing `CALL sa_conn_info( )`.

To filter according to a database:

```
CALL sa_server_option( 'RequestFilterDB' , database-id );
```

The `database-id` can be obtained by executing `SELECT CONNECTION_PROPERTY( 'DBNumber' )` when connected to that database. Filtering continues until explicitly reset, or until the database server is shut down.

To reset filtering, use either of the following two statements to reset filtering either by connection or by database:

```
CALL sa_server_option( 'RequestFilterConn' , -1 );
```

```
CALL sa_server_option( 'RequestFilterDB' , -1 );
```

To include host variable values in the request log:

- use the -zr server option with a value of hostvars
- execute the following:

  ```
  CALL sa_server_option( 'RequestLogging' , 'hostvars' );
  ```

The request log analysis procedure, sa_get_request_times, recognizes host variables in the log and adds them to the global temporary table SATMP_request_hostvar.

1.8.4.2.2 The Timing Utilities

Some performance testing utilities, including fetchtst, instest, and trantest, are available in *%SQLANYSAMP17%*\SQLAnywhere.

The fetchtst utility measures fetch rates for an arbitrary query. The instest utility determines the time required for rows to be inserted into a table. The trantest utility measures the load that can be handled by a given server configuration given a database design and a set of transactions.

These tools give you more accurate timings than the graphical plan with statistics, and can provide an indication of the best achievable performance (for example, throughput) for a given server and database configuration.

Complete documentation for the tools can be found in the readme.txt file in the same folder as the utility.

1.8.4.2.3 Database Performance Monitoring

Monitor your database performance in real time.

A set of statistics that you can use to monitor database performance is provided. There are many ways to access these statistics:

SQL functions

These functions allow your application to access database statistics directly.

Microsoft Windows Performance Monitor

This is a monitoring tool provided by your Microsoft Windows operating system. You can adjust what it monitors with database server options.

Performance Statistics utility for UNIX and Linux (dbstats)

This utility provides monitoring of database server, database, and connection statistics for database servers running on UNIX and Linux.

In this section:

SQL Functions Used to Monitor Statistics [page 1504]
A set of system functions that can provide data on a per-connection, per-database, or server-wide basis is provided.

Windows Performance Monitor [page 1506]
You can use also use the Windows Performance Monitor to monitor a database server.

1.8.4.2.3.1 SQL Functions Used to Monitor Statistics

A set of system functions that can provide data on a per-connection, per-database, or server-wide basis is provided.

The kind of information available ranges from static information (such as the database server name) to detailed performance-related statistics (such as disk and memory usage).

Functions That Retrieve System Information

The following functions retrieve system information:

PROPERTY function

This function provides the value of a given property on a server-wide basis.

DB_PROPERTY and DB_EXTENDED_PROPERTY functions

These functions provide the value of a given property for a given database, or by default, for the current database.

CONNECTION_PROPERTY and CONNECTION_EXTENDED_PROPERTY functions

These functions provide the value of a given property for a given connection, or by default, for the current connection.

Supply as an argument only the name of the property you want to retrieve. The functions return the value for the current server, connection, or database.

Improving Query Efficiency

For better performance, a client application monitoring database activity should use the PROPERTY_NUMBER function to identify a named property, and then use the number to repeatedly retrieve the statistic.

Property names obtained in this way are available for many different database statistics, from the number of transaction log page write operations and the number of checkpoints performed, to the number of reads of index leaf pages from the memory cache.

The following set of statements illustrates the process from Interactive SQL:

```
CREATE VARIABLE propnum INT;
CREATE VARIABLE propval INT;
SET propnum = PROPERTY_NUMBER( 'CacheRead' );
SET propval = DB_PROPERTY( propnum );
```

Example

The following statement sets a variable named server_name to the name of the current server:

```
SET server_name = PROPERTY( 'name' );
```

The following query returns the user ID for the current connection:

```
SELECT CONNECTION_PROPERTY( 'UserID' );
```

The following query returns the file name for the root file of the current database:

```
SELECT DB_PROPERTY( 'file' );
```

1.8.4.2.3.2 Windows Performance Monitor

You can use also use the Windows Performance Monitor to monitor a database server.

The Windows Performance Monitor (PerfMon) offers performance statistics including network communication statistics. It uses a shared-memory scheme for performing queries against the database server, so it does not affect the statistics themselves.

The you can specify the database server options, and the maximum number of connections or databases that the Performance Monitor can monitor with the -ks, -ksc, and -ksd database server options.

You can only monitor one database server per instance of the Windows Performance monitor.

1.8.4.2.4 Procedure Profiling (System Procedures)

The Profiler is the preferred method for profiling stored SQL objects. But you can also use can use system procedures to view procedure profiling information for stored procedures, functions, events, system triggers, and triggers.

Perform procedure profiling by enabling procedure profiling and executing these system procedures:.

sa_procedure_profile

The sa_procedure_profile system procedure shows in-depth profiling information, including execution times for the lines within each object; each line in the result set represents an executable line of code in the object.

sa_procedure_profile_summary

The sa_procedure_profile_summary system procedure shows you the overall execution time for each object, giving you a summary of all objects that ran; each line in the result set represents the execution details for one object.

When reviewing the results from these system procedures, there may be more objects listed than those specifically called. This is because one object can call another object. For example, a trigger might call a stored procedure that, in turn, calls another stored procedure. Some procedures might not be listed if they are simple enough to have been inlined.

In this section:

1.8.4.2.4.1 Enabling Procedure Profiling (SQL)

Enable procedure profiling using the sa_server_option system procedure.

Prerequisites

You must have the DIAGNOSTICS system role, and the MANAGE PROFILING system privilege.

Procedure

1. Connect to the database.
2. Call the sa_server_option system procedure, setting the ProcedureProfiling option to ON.

   For example, execute the following CALL statement:

   ```
   CALL sa_server_option( 'ProcedureProfiling' , 'ON' );
   ```

Results

Procedure profiling is enabled.

1.8.4.2.4.2 Filtering Procedure Profiling by User (SQL)

View the procedures that a specific user is using without preventing other connections from using the database.

Prerequisites

You must have the DIAGNOSTICS system role, and the MANAGE PROFILING system privilege.

Context

This is useful if the connection already exists, or if multiple users connect with the same user ID.

Procedure

1. Connect to the database.
2. Call the sa_server_option system procedure as follows:

```
CALL sa_server_option( 'ProfileFilterUser' , 'userid' );
```

Results

The value of `userid` is that of the user being monitored.

1.8.4.2.4.3 Resetting Procedure Profiling (SQL)

Reset procedure profiling using the sa_server_option system procedure.

Prerequisites

You must have the DIAGNOSTICS system role, and the MANAGE PROFILING system privilege.

Procedure profiling must be enabled.

Context

When you reset profiling, the database server clears the profiling history and immediately starts collecting new information about procedures, functions, events, and triggers.

Procedure

To clear profiling history and restart data collection, call the sa_server_option system procedure and set the ProcedureProfiling option to RESET.

```
CALL sa_server_option( 'ProcedureProfiling' , 'RESET' );
```

Results

Procedure profiling is reset.

1.8.4.2.4.4 Disabling Procedure Profiling (SQL)

Disable procedure profiling and optionally clear profiling history using the sa_server_option system procedure.

Prerequisites

You must have the DIAGNOSTICS system role, and the MANAGE PROFILING system privilege.

Context

Once you are finished, you can clear the profiling history and disable profiling (using the CLEAR option), or just temporarily disable profiling (OFF option). In either case, the database server stops collecting profiling information.

Procedure

To stop data collection and clear profiling history, call the sa_server_option system procedure and set the ProcedureProfiling option to CLEAR.

```
CALL sa_server_option( 'ProcedureProfiling' , 'CLEAR' );
```

To stop data collection and retain profiling history, call the sa_server_option system procedure and set the ProcedureProfiling option to OFF.

```
CALL sa_server_option( 'ProcedureProfiling' , 'OFF' );
```

Results

Profiling history is retained or cleared and procedure profiling is disabled.

1.8.4.2.5 Diagnostic Tracing (Deprecated)

The Diagnostic tracing feature is deprecated. Use the SQL Anywhere Profiler to diagnose issues in your database.

The diagnostic tracing data produced by the database server can include the timestamps and connection IDs of statements handled by the database server. For queries, diagnostic tracing data includes the isolation level, number of rows fetched, cursor type, and query execution plan. For INSERT, UPDATE, and DELETE statements, the number of rows affected is also included. You can also use diagnostic tracing to record information about locking and deadlocks, and to capture performance statistics.

You can use the data gathered during diagnostic tracing to perform in-depth application profiling activities, such as identifying and troubleshooting the following

- specific performance problems
- statements that are unusually slow to execute
- improper option settings
- circumstances that cause the optimizer to pick a sub-optimal plan
- contention for resources (CPUs, memory, disk I/O)
- application logic problems

The tracing architecture is robust and scalable. It can record all the information that request logging records, as well as details to support tailored analysis.

> **i** Note
>
> Diagnostic tracing does not capture an event, but captures long-running statements within the event. To capture an event, embed the event code within a procedure and call the procedure from the event. Before running the procedure, turn on tracing with a custom level that specifies low detail with no conditions on statement capture.

In this section:

1.8.4.2.5.1 Trace Session Data (Deprecated)

Diagnostic tracing data is gathered during a **tracing session**.

Use the ATTACH TRACING and DETACH TRACING statements to capture tracing session data:

When a tracing session is in progress, the database server generates diagnostic information for the specified database. The amount of tracing data generated depends on the tracing settings.

The database being profiled is either referred to as the **production database**, the source database, or the database being profiled. The database into which the tracing data is stored is referred to as the **tracing database**. The production and tracing database can be the same database. However, to avoid increasing the size of the production database, store tracing data in a separate database. The size of database files cannot be reduced after they have grown. Also, the production database performs better if the overhead for storing and maintaining tracing data is performed in another database, especially if the production database is large and heavily used.

The tables in the tracing database that hold the tracing data are referred to as the **diagnostic tracing tables**. These tables are owned by dbo.

> **i** Note
>
> Tracing information is *not* unloaded as part of a database unload or reload operation. To transfer tracing information from one database to another you must manually copy the contents of the sa_diagnostic_* tables; however, this is not recommended.

1.8.4.2.5.2 Diagnostic Tracing Configuration (Deprecated)

You can use system procedures to change settings stored in the diagnostic tracing tables.

The diagnostic tracing feature is deprecated. Use the SQL Anywhere Profiler to diagnose issues in your database.

Tracing settings are stored in the sa_diagnostic_tracing_level system table.

The SendingTracingTo and ReceivingTracingFrom database properties identify the tracing and production databases, respectively.

1.8.4.2.5.3 Diagnostic Tracing Levels (Deprecated)

Set the tracing level using the sa_set_tracing_level system procedure.

To see the current tracing levels set for a database, look in the sa_diagnostic_tracing_level table.

Estimated impacts to performance reflect the assumption that tracing data is sent to a tracing database on another database server (recommended).

The following is a list of diagnostic tracing levels..

Level 0

This level keeps the tracing session running, but does not send any tracing data to the tracing tables.

Level 1

Performance counters and a sampling of executed statements (once every five seconds) are gathered. For this level, the diagnostic tracing types include:

- volatile_statistics, with sampling every 1 second
- nonvolatile_statistics, with sampling every 60 seconds

This level has a negligible impact on performance.

Level 2

This level gathers performance counters, a sampling of executed plans (once every five seconds), and records all executed statements. For this level, the diagnostic tracing types include:

- volatile_statistics, with sampling every 1 second
- nonvolatile_statistics, with sampling every 60 seconds
- statements
- plans, sampling every 5 seconds

This level has a medium impact on performance (up to, but not more than, a 20% overhead).

Level 3

This level records the same details as Level 2 but with more frequent plan samples (once every 2 seconds) and detailed blocking and deadlock information. For this level, the diagnostic tracing types include:

- volatile_statistics, with sampling every 1 second
- nonvolatile_statistics, with sampling every 60 seconds
- statements
- blocking
- deadlock
- statements_with_variables
- plans, with sampling every 2 seconds

This level has the greatest impact on performance (greater than 20% overhead).

1.8.4.2.5.4 Diagnostic Tracing Scopes (Deprecated)

Scope values can be used to limit tracing to who (or what) is causing the activity in the database.

Use the sa_set_tracing_level system procedure values.

The following is the list of **scopes** for diagnostic tracing. For example, you can set the scope to trace requests coming from a specified connection. Scope values are stored in the scope column of the dbo.sa_diagnostic_tracing_level diagnostic table, and may have corresponding arguments, typically an identifier such as an object name or user name, which are stored in the identifier column.

Values in the scope column	Description
DATABASE	Records tracing data for any event occurring within the database, assuming the event corresponds to the specified level and condition. Used for long-term background monitoring of the database, or for short-term diagnostics, when it is necessary to determine the source of costly queries. There is no identifier to specify when you specify DATABASE.
ORIGIN	Records tracing data for the queries originating from either outside or inside the database. There are two possible identifiers you can specify when specifying the scope ORIGIN: External or Internal. External specifies to log the statement text and associated details for queries that come from outside the database server, and that correspond to the specified level and condition. Internal specifies to log the same information for queries that come from within the database server, and that correspond to the level and condition specified.
USER	Records tracing data only for the queries issued by the specified user, and by connections created by the specified user. This scope is used to diagnose problematic queries originating from a particular user. The identifier for this scope is the user ID of the user for whom the tracing is to be performed.
CONNECTION_NAME, or CONNECTION_NUMBER	Records tracing data only for the statements executed by the current connection. These scopes are used when the user has multiple connections, one of which is executing costly statements. The identifier for this scope is the name of the connection, or the connection number, respectively.

Values in the scope column	Description
FUNCTION, PROCEDURE, EVENT, TRIGGER, or TABLE	Records tracing data for the statements that use the specified object. If the object references other objects, all the data for those objects is recorded as well. For example, if tracing is being done for a procedure that uses a function which, in turn, triggers an event, statements for all three objects are logged, providing they correspond to the specified level and condition provided for logging. Used when use of a specific object is costly, or when the statements that reference the object take an unusually long time to finish. The TABLE scope is used for tables, materialized views, and non-materialized views. The identifier for this scope is the fully qualified name of the object.

1.8.4.2.5.5 Customized Diagnostic Tracing Levels (Deprecated)

Diagnostic tracing settings are grouped into several levels, but you can also customize the settings further within these levels, which are referred to as **diagnostic tracing types**.

Customizing diagnostic tracing settings allows you to reduce the amount of unwanted tracing data in the diagnostic tracing session. For example, suppose that user AliceB has been complaining that her application has been running slowly, yet the rest of the users are not experiencing the same problem. You now want to know exactly what is going on with AliceB's queries. Gather the list of all queries and other statements that AliceB runs as part of her application, and any query plans for long running queries. To do this, you could set the diagnostic tracing level to 3 and generate tracing data for a day or two. However, since this level can significantly impact performance for other users, it would be better to limit the tracing to just AliceB's activities. To do this, set the diagnostic tracing level to 3, and then customize the scope of the diagnostic tracing to be USER, and specify AliceB as the user name. Allow the diagnostic tracing session to run for a couple of hours, and then examine the results.

Use the sa_set_tracing_level system procedure to customize your diagnostic tracing settings.

Do not change diagnostic tracing settings while a tracing session is in progress because it makes interpreting the data more difficult.

1.8.4.2.5.6 Diagnostic Tracing Types (Deprecated)

There are several **diagnostic tracing types** you can choose for diagnostic tracing.

Each diagnostic tracing type requires a corresponding condition, as noted below, and is stored in the trace_type column of the dbo.sa_diagnostic_tracing_level diagnostic table, and may have corresponding diagnostic tracing conditions, which are stored in the trace_condition column.

Value in the trace_type column	Description
VOLATILE_STATISTICS	Collects a sample of frequently changing database and server statistics. Scopes and conditions: This diagnostic tracing type requires the DATABASE scope, and uses the SAMPLE_EVERY condition as the interval at which to collect the data.
NONVOLATILE_STATISTICS	Collects a sample of database and server statistics that do not change frequently. Non-volatile statistics cannot be collected more frequently than volatile statistics. Volatile statistics must be collected in order for non-volatile statistics to be collected, and the time difference between the sampling for non-volatile statistics should be a multiple of the time difference specified for the volatile statistics. Scopes and conditions: This diagnostic tracing type requires the DATABASE scope, and uses the SAMPLE_EVERY condition as the interval at which to collect the data.
CONNECTION_STATISTICS	Collects a sample of connection statistics. If the scope is database, statistics for all connections to the database are collected. If the scope is user, statistics for all connections for the specified user are collected. If the scope is CONNECTION_NAME or CONNECTION_NUMBER, only statistics for the specified connection are collected. Volatile statistics have to be collected in order for CONNECTION_STATISTICS to be collected, and the time interval between sampling should be a multiple of that specified for the VOLATILE_STATISTICS. Scopes and conditions: This diagnostic tracing type can be used with the DATABASE, USER, CONNECTION_NUMBER, and CONNECTION_NAME scopes, and uses the SAMPLE_EVERY condition as the interval at which to collect the data.
BLOCKING	Collects information about blocks according to the specified scope and condition. If the scope is CONNECTION_NAME or CONNECTION_NUMBER, then the block may be recorded when the connection blocks another connection, or is blocked by another connection. Scopes and conditions: This diagnostic tracing type can be used with all the scopes, and can use any one of the following conditions for collection: NONE, NULL, SAMPLE_EVERY.

Value in the trace_type column	Description
PLANS	Collects execution plans for queries, depending on the condition and scope. Scopes and conditions: This diagnostic tracing type can be used with all the scopes, and can use any one of the following conditions for collection: NONE, NULL, SAMPLE_EVERY, and ABSOLUTE_COST.
PLANS_WITH_STATISTICS	Collects plans with execution statistics. Plans are recorded at cursor close time. If the RELATIVE_COST_DIFFERENCE condition is specified, part of the statistics in the output might be best-guess statistics. Scopes and conditions: This diagnostic tracing type can be used with all the scopes, and accepts any one of the conditions for collection.
STATEMENTS	Collects SQL statements for the specified scope and condition. Internal variables are collected the first time each procedure is executed. This diagnostic tracing type is automatically included if the STATEMENTS_WITH_VARIABLES, or PLANS, PLANS_WITH_STATISTICS diagnostic tracing type is specified. Scopes and conditions: This diagnostic tracing type can be used with all the scopes, and can use any one of the conditions for collection.
STATEMENTS_WITH_VARIABLES	Collects SQL statements and the variables attached to the statements. For each variable, either internal or host, all the values that were assigned are collected as well. Scopes and conditions: This diagnostic tracing type can be used with all the scopes, and can use any one of the conditions for collection.
OPTIMIZATION_LOGGING	This diagnostic tracing type is deprecated. If you specify this trace type, the software ignores it.
OPTIMIZATION_LOGGING_WITH_PLANS	This diagnostic tracing type is deprecated. If you specify this trace type, the software ignores it.

1.8.4.2.5.7 Diagnostic Tracing Conditions (Deprecated)

Conditions must be met in order for a tracing entry to be made for a specific diagnostic tracing type.

The following table lists the diagnostic tracing **conditions** you can set. Most conditions require a value, as noted below. Conditions are stored in the trace_condition column of the dbo.sa_diagnostic_tracing_level

diagnostic table, and may have a corresponding value, such as an amount of time in milliseconds, stored in the value column.

Value in the trace_condition column	Description
NONE, or NULL	Records all the tracing data that satisfies the level and scope requirements. Using expensive diagnostic tracing levels (plans, for example) with this condition for extended time periods is not recommended.
SAMPLE_EVERY	Records tracing data that satisfies the level and scope requirements if more than the specified time interval has elapsed since the last event was recorded. Values: This condition takes a positive integer, reflecting time in milliseconds.
ABSOLUTE_COST	Records the statements with cost of execution greater than, or equal to, the specified value. Values: This condition takes a cost value, specified in milliseconds.
RELATIVE_COST_DIFFERENCE	Records the statements for which the difference between the expected time for execution and the real time for execution is greater than or equal to the specified value. Values: This condition takes a cost value specified as a percentage. For example, to log statements that are at least twice as slow as estimated, specify a value of 200.

1.8.4.2.5.8 Determining Current Diagnostic Tracing Settings (SQL) (Deprecated)

Retrieve the diagnostic tracing settings in effect by querying the sa_diagnostic_tracing_level table.

Prerequisites

You must have the DIAGNOSTICS system role, and the MANAGE PROFILING system privilege.

Procedure

1. Connect to the database.
2. Query the sa_diagnostic_tracing_level table for rows in which the enabled column contains a 1.

Results

The database server returns the diagnostic tracing settings currently in use. A 1 in the enabled column indicates that the setting is in effect.

Example

The following statement shows you how to query the sa_diagnostic_tracing_level diagnostic table to retrieve the current diagnostic tracing settings:

```
SELECT * FROM sa_diagnostic_tracing_level WHERE enabled = 1;
```

The following table is an example result set from the query:

id	scope	identifier	trace_type	trace_condition	value	enabled
1	database	(NULL)	volatile_statistics	sample_every	1,000	1
2	database	(NULL)	nonvolatile_statistics	sample_every	60.000	1
3	database	(NULL)	connection_statistics	(NULL)	60,000	1
4	database	(NULL)	blocking	(NULL)	(NULL)	1
5	database	(NULL)	deadlock	(NULL)	(NULL)	1
6	database	(NULL)	plans_with_statistics	sample_every	2,000	1

1.8.4.2.5.9 Diagnostic Tracing Configuration Settings (Deprecated)

Diagnostic tracing settings are specific to a database.

Use the sa_set_tracing_level system procedure to change the diagnostic tracing level. This procedure does not start a tracing session and fails if a tracing session is already in progress.

Example

The following statement uses the sa_set_tracing_level system procedure to set the diagnostic tracing level to 1:

```
CALL sa_set_tracing_level( 1 );
```

Existing settings are overwritten with the default settings associated with diagnostic tracing level 1.

1.8.4.2.5.10 Creating a Diagnostic Tracing Session (SQL) (Deprecated)

Start a tracing session by executing the ATTACH TRACING statement in Interactive SQL.

Prerequisites

You must have the DIAGNOSTICS system role, and the MANAGE PROFILING system privilege.

Context

Starting a tracing session is also referred to as attaching tracing. Likewise, stopping a tracing session is referred to as detaching tracing. The SQL statements for starting and stopping tracing are, respectively, ATTACH TRACING and DETACH TRACING.

Procedure

1. Connect to the database.
2. Use the sa_set_tracing_level system procedure to set the tracing levels. For example:

   ```
   CALL sa_set_tracing_level( 1 );
   ```
3. Start tracing by executing an ATTACH TRACING statement.
4. Stop tracing by executing a DETACH TRACING statement.

Results

The tracing session is created and completed.

Example

This example shows how to start diagnostic tracing on the current database, store the tracing data in a separate database, and set a two hour limit on the amount of data to store. This example assumes there is a user ID DBA with password passwd with the correct privileges:

```
ATTACH TRACING TO 'UID=DBA;PWD=passwd;Server=server47;DBN=tracing;Host=myhost'
LIMIT HISTORY 2 HOURS;
```

This example shows how to start diagnostic tracing on the current database, store the tracing data in the local database, and set a two megabyte limit on the amount of data to store:

```
ATTACH TRACING TO LOCAL DATABASE LIMIT SIZE 2 MB;
```

This example shows how to stop diagnostic tracing and save the diagnostic data that was captured during the tracing session:

```
DETACH TRACING WITH SAVE;
```

This example shows how to stop diagnostic tracing and not save the diagnostic data.

```
DETACH TRACING WITHOUT SAVE;
```

1.8.4.2.5.11 Analysis of Diagnostic Tracing Information (Deprecated)

Diagnostic tracing data provides a record of all activities that took place on the database server and that correspond to the diagnostic tracing levels and the tracing session settings.

When reviewing the data, you must consider the settings that were in place. For example, the absence of a statement that you expected to see in a tracing session might indicate that the statement never ran, but it might also indicate that the statement was not expensive enough to fulfill a condition that only expensive statements be traced.

There are many reasons to examine in detail what activities the database server is performing. These include troubleshooting performance problems, estimating resource usage to plan for future workloads, and debugging application logic.

> **i** Note
>
> Diagnostic tracing does not capture an event, but captures long-running statements within the event. To capture an event, embed the event code within a procedure and call the procedure from the event. Before running the procedure, turn on tracing with a custom level that specifies low detail with no conditions on statement capture.

1.8.4.2.5.12 Creating an External Tracing Database (Command Line) (Deprecated)

Use the Unload utility (dbunload) to manually create a tracing database without a tracing session.

Prerequisites

You must have the DIAGNOSTICS system role, and the MANAGE PROFILING system privilege.

Procedure

1. Connect to the database.
2. Run a dbunload command to unload the schema from the production database into the new tracing database:

   For example:

   ```
   dbunload -c "UID=DBA;PWD=sql;Server=demo;DBN=demo" -an tracing.db -n -kd
   ```

   This example creates a new database with the name supplied by the -an option (`tracing.db`). The -n option unloads the schema from the database being profiled (in this case, the SQL Anywhere sample database, *demo.db*) into the new tracing database. The -kd option places all the dbspaces in a single dbspace file.
3. To store the tracing database on a separate computer, copy it to the new location.

Results

An external database to store analysis data is created and a tracing session is not created.

1.8.4.2.6 Monitoring Database Servers (Windows Performance Monitor)

Use the Windows Performance Monitor to view counters related to your database, server, or connection.

Prerequisites

Consult your Windows operating system documentation to ensure that you have the required permissions and setup to run the Windows Performance Monitor.

Context

The Windows Performance Monitor (PerfMon) uses a shared-memory scheme for performing queries against the database server, so it does not affect the statistics themselves.

You can control the maximum number of connections or databases that the Performance Monitor can monitor with the -ksc, and -ksd database server options.

The -k and -ks database server options prevent the Windows Monitor from being able to monitor a database server.

You can monitor one database server per instance of the Windows Performance Monitor.

Procedure

1. Start the database server that you want to monitor and ensure that it can accept shared memory connections. If you need to log statistics, then you must start the database server as a service.
2. Start the Performance Monitor.
   Run the following command from the command line:`perfmon`.
3. In the navigation tree, click *Performance Monitor*.
4. Add counters.
   a. In the right pane, click the Plus sign tool (+) on the toolbar.
   b. In the *Performance Object* list, select one of the following:

      SQL Anywhere 17 Connection

      This counter monitors performance for a single connection. A connection must currently exist to see this selection.

      SQL Anywhere 17 Database

      This counter monitors performance for a single database.

      SQL Anywhere 17 Server

      This counter monitors performance on a server-wide basis.
5. In the *Counter* list, click a statistic to view.
   If you clicked *SQL Anywhere 17 Connection* or *SQL Anywhere 17 Database*, choose a database connection or database to monitor from the *Instances* box.
6. To display the counter, click *Add*.
7. When you have selected all the counters you want to display, click *Close*.

Results

The specified statistics are displayed in the Windows Performance Monitor. Refer to the Windows Performance Monitor documentation for more information.

1.8.4.2.7 Identifying and Fixing Index Fragmentation (SQL)

Use the sa_index_density stored procedure to identify index fragmentation, and then rebuild the indexes to fix the problem.

Prerequisites

Ensure that there are no other connections to the database.

To execute the sa_index_density system procedure, you must have EXECUTE privilege on the system procedure, as well as one of the following system privileges:

MONITOR
MANAGE ANY STATISTICS
CREATE ANY INDEX
ALTER ANY INDEX
DROP ANY INDEX
CREATE ANY OBJECT
ALTER ANY OBJECT
DROP ANY OBJECT

To alter an index on a table, you must be the owner of the table, or have one of the following privileges:

REFERENCES privilege on the table
ALTER ANY INDEX system privilege
ALTER ANY OBJECT system privilege

Context

Periodically check for fragmentation on your production database. When an index is created, table data is read and values for the index are recorded on index pages following a logical order. As data changes in the table, new index values can be inserted between existing values. To maintain the logical order of index values, the database server may have to create new index pages to accommodate existing values that are moved. The new pages are not usually adjacent to the pages on which the values were originally stored. This cumulative degradation in the order of index pages is called index fragmentation.

Commonly executed queries taking longer to perform on tables where large blocks of rows are continuously being inserted, updated, and deleted is a symptom of index fragmentation.

Procedure

1. In Interactive SQL, connect to your database and execute a statement similar to the one below to test the index density on a specific table:

   ```
   CALL sa_index_density( 'table-name' );
   ```

   Density values range between 0 and 1. Values closer to 1 indicate little index fragmentation. Values less than 0.5 indicate a level of index fragmentation that can impact performance.
2. Execute the following ALTER INDEX...REBUILD statement to improve the density of an index:

   ```
   ALTER INDEX PRIMARY KEY ON table-name REBUILD;
   ```
3. Close Interactive SQL.

Results

You have used Interactive SQL to identify and fix index fragmentation.

1.8.4.2.8 Identifying and Fixing Table Fragmentation (SQL)

Determine if your database has table fragmentation, and if necessary, learn how to fix table fragmentation

Prerequisites

To run the sa_table_fragmentation system procedure, you must have the EXECUTE privilege on the system procedure, as well as the MANAGE ANY STATISTICS or MONITOR system privilege.

To run the REORGANIZE TABLE statement, you must be the owner of the table, or have the REORGANIZE ANY OBJECT system privilege.

Context

Commonly executed queries taking longer to perform on tables where large blocks of rows are continuously being inserted, updated, and deleted is a symptom of index fragmentation.

Table data is stored on database pages. When data manipulation statements such as INSERT, UPDATE, and DELETE are executed against a table, rows might not be stored contiguously, or might be split between multiple pages. Even though CPU activity is high, table fragmentation can negatively impact the performance of queries that require a scan of the table.

Procedure

1. In Interactive SQL, connect to your database and execute a statement similar to the one below to test for table fragmentation:

   ```
   CALL sa_table_fragmentation( 'table-name' );
   ```

   If the value in the segs_per_row (the number of segments per row) column is greater than 1.1, then table fragmentation is present. Higher degrees of fragmentation may negatively impact performance.
2. Execute a REORGANIZE TABLE statement to reduce table fragmentation:

   ```
   REORGANIZE TABLE table-name;
   ```
3. Close Interactive SQL.

Results

You have used diagnosed table fragmentation and reorganized the table to fix it.

1.9 User and Database Security

Many features are provided to help you prevent unauthorized access to data, and unauthorized activities on the database.

In this section:

1.9.1 User Security (Roles and Privileges)

A **role-based access control model** (RBAC) is provided for the execution of privileged operations.

A role-based security model provides complete control and granularity for the privileges you want to grant to users. Each privileged operation a user can perform in the database requires one or more system or object-level privileges.

A **system privilege** is a right to perform an authorized database task. For example, the CREATE TABLE system privilege allows a user to create self-owned tables.

An **object-level privilege** is a right to perform an authorized task on a specified object. For example, having ALTER privileges on TableA allows a user to alter that table, but not other tables.

A **role** is a collection of one or more system privileges, object-level privileges, or roles. You can grant roles to other roles to create a role hierarchy. Granting a role to a user is equivalent to granting the user the underlying system privileges for the role.

Each new or migrated database includes a predefined set of roles you can use to get started. These system roles act as a starting point for implementing role-based security.

In this section:

Tutorial: Granting Roles and Privileges (SQL) [page 1649]
In the role-based security model, users must have the correct privileges and roles to perform specific database operations.

Upgrading from Authority-based Security to Role-based Security [page 1657]
You can upgrade your database from authority-based security to role-based security.

Related Information

Upgrading from Authority-based Security to Role-based Security [page 1657]

1.9.1.1 Roles

There are three types of roles in the role-based security model: **system roles**, **user-defined roles** (which include user-extended roles), and **compatibility roles**.

View the roles and privileges a user has in SQL Central by clicking the user and viewing the details that are displayed. You can also retrieve the details using the sp_displayroles system procedure.

In this section:

System Roles [page 1528]
System roles contain the set of privileges required for more complex operations.

Compatibility Roles [page 1532]
Compatibility roles can be thought of as starter roles containing logical groups of privileges.

User-defined Roles [page 1543]
A user-defined role is a collection you can create of system privileges, object-level privileges, and roles, typically created to group privileges related to a specific task or set of tasks.

User-extended Roles [page 1548]
A **user-extended role** is a user that has been extended to be a role, and is a type of user-defined role.

Role Administrators [page 1555]
Role administrators are responsible for granting and revoking user-defined roles to users and other roles.

System Privileges Introduced in Upgrades [page 1558]
A new release of the software may introduce new system privileges.

Configuring Roles and Privileges For a Role (SQL Central) [page 1561]
Grant roles or privileges to any role, revoke roles or privileges from it, and set the administrative rights the role has over the roles and privileges it has been granted.

Configuring Roles and Privileges For a Role (SQL) [page 1562]
Grant roles or privileges to any role, revoke roles or privileges from it, and set the administrative rights the role has over the roles and privileges it has been granted.

Granting a Role (SQL Central) [page 1563]
Grant a role to a user or another role.

Granting a Role (SQL) [page 1565]
Grant a role to a user or another role.

Revoking a Role (SQL Central) [page 1566]
Revoke a role from a user, a user-extended role, or a user-defined role.

Revoking a Role (SQL) [page 1568]
Revoke a role from a user, a user-extended role, or a user-defined role.

Dropping a Role (SQL Central) [page 1569]
Drop a user-defined, user-extended, or compatibility role.

Dropping a Role (SQL) [page 1570]
Drop a user-defined, user-extended, or compatibility role.

Related Information

Upgrading from Authority-based Security to Role-based Security [page 1657]
sp_displayroles System Procedure

1.9.1.1.1 System Roles

System roles contain the set of privileges required for more complex operations.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Here are a few points to remember about system roles.

- You cannot drop system roles.
- You cannot grant administrative rights (WITH ADMIN OPTION or WITH ADMIN ONLY OPTION) when granting system roles. Administrative rights on these system roles lies solely with MANAGE ROLES system privilege.
- When a system role is granted to a user-extended role, grantees of the user-extended role inherit the system role as well.
- With the exception of the SYS role, you can grant/revoke additional privileges and roles to/from a system role, provided you have administrative rights on the privileges and roles you are granting/revoking.
- With the exception of the SYS, dbo, and rs_systabgroup role, system roles do not own objects.

System role	Description
dbo	This role owns many system stored procedures and views, tables used for UltraLite and MobiLink, and is a grantee of the SYS, SYS_AUTH_DBA_ROLE, and SYS_AUTH_RESOURCE_ROLE roles. Only users with the MANAGE ROLES system privilege can administer this role.
DIAGNOSTICS	This role grants SELECT, INSERT, UPDATE, DELETE, and ALTER privileges on diagnostic tables and views. Only users with the MANAGE ROLES system privilege can administer this role.
PUBLIC	This role has SELECT privilege on the system tables. As well, the PUBLIC role is a grantee of the SYS and dbo roles, and has read access for some of the system tables and views so users can find out information about the database schema.
rs_systabgroup	This role owns tables and system procedures that are required for Replication Server. Only users with the MANAGE ROLES system privilege can administer this role.
SA_DEBUG	This role is required for the SQL Anywhere Debugger. Only users with the MANAGE ROLES system privilege can administer this role.
SYS	This role owns the catalog, which contains the full description of the database schema, including all database objects and all user IDs. You cannot grant or revoke additional privileges to or from this role. Only users with the MANAGE ROLES system privilege can administer this role.
SYS_OFFLINE_RESET_PASSWORD_ROLE	This role is required to change the password for a database user. Only users with the MANAGE ROLES system privilege can administer this role.
SYS_RUN_PROFILER_ROLE	This role is required for the SQL Anywhere Profiler. It is also required to view the statement performance summary. It contains the privileges required for the diagnostic and troubleshooting tasks within this application. Only users with the MANAGE ROLES system privilege can administer this role.
SYS_REPLICATION_ADMIN_ROLE	This role is required for performing administration tasks related to replication such as granting replication roles, managing publications, subscriptions, synchronization users and profiles, managing message types, setting replication-related options, and so on. Only users with the MANAGE REPLICATION system privilege can administer this role.
SYS_RUN_REPLICATION_ROLE	This role is required for performing replication using the dbremote utility, and performing synchronization using the dbmlsync utility. Only users with the SYS_REPLICATION_ADMIN_ROLE system role can administer this role.
SYS_SAMONITOR_ADMIN_ROLE	This role is required for the SQL Anywhere Monitor.

System role	Description
SYS_SPATIAL_ADMIN_ROLE	This role allows users to create, alter, or drop spatial reference systems and spatial units of measure. Only users with the MANAGE ROLES system privilege can administer this role.

In this section:

1.9.1.1.1.1 Replication-related System Roles

There are two system roles related to replication: SYS_RUN_REPLICATION_ROLE, and SYS_REPLICATION_ADMIN_ROLE.

In this section:

Related Information

Compatibility Roles [page 1532]
sp_has_role System Procedure

1.9.1.1.1.1.1 The SYS_REPLICATION_ADMIN_ROLE System Role

The SYS_REPLICATION_ADMIN_ROLE system role is required for performing administrative tasks related to replication such as granting replication roles, managing publications, subscriptions, synchronization users and profiles, managing message types, and setting replication-related options.

The system privileges granted to SYS_REPLICATION_ADMIN_ROLE are:

- MANAGE REPLICATION

- SET ANY SYSTEM OPTION
- SET ANY PUBLIC OPTION
- SET ANY USER DEFINED OPTION
- SELECT ANY TABLE
- CREATE ANY PROCEDURE
- DROP ANY PROCEDURE
- MANAGE ANY WEB SERVICE
- CREATE ANY TABLE
- DROP ANY TABLE
- SERVER OPERATOR
- MANAGE ANY USER
- MANAGE ROLES
- MANAGE ANY OBJECT PRIVILEGE

These are the minimum (and irrevocable) privileges required for these two roles, but you can grant additional privileges as well.

In addition to the statements associated with the privileges above, users with the SYS_REPLICATION_ADMIN_ROLE system role can execute the following statements:

- SYNCHRONIZE PROFILE statement
- REMOTE RESET statement
- LOCK FEATURE statement

Related Information

sp_has_role System Procedure
GRANT ROLE Statement

1.9.1.1.1.1.2 The SYS_RUN_REPLICATION_ROLE System Role

The SYS_RUN_REPLICATION_ROLE system role is required for performing replication using the dbremote utility and performing synchronization using the dbmlsync utility.

The system privileges granted to SYS_RUN_REPLICATION_ROLE are:

- MONITOR
- BACKUP DATABASE
- DROP CONNECTION
- SELECT ANY TABLE
- SET ANY SYSTEM OPTION
- SET ANY USER DEFINED OPTION
- ACCESS USER PASSWORD

The SYS_RUN_REPLICATION ROLE system role also inherits the system privileges granted to the SYS_AUTH_DBA_ROLE compatibility role.

In addition to the statements associated with the privileges above, users with the SYS_RUN_REPLICATION_ROLE system role can execute the SYNCHRONIZE statement.

In MobiLink, when the connection is made from the SQL Anywhere synchronization client (dbmlsync) utility, the SYS_RUN_REPLICATION_ROLE system role enables dbmlsync to have full access to the database. Any other connection using the same user ID is granted no special authority.

In SQL Remote, when the connection is made from the Message Agent, the SYS_RUN_REPLICATION_ROLE system role enables the Message Agent to have full access to the database to make any changes contained in the messages. Any other connection using the same user ID is granted no special authority.

When a connection is made to the database in any other way, the user cannot exercise any of the privileges associated with this role. For example, the user cannot use the SELECT ANY TABLE system privilege.

The SYS_RUN_REPLICATION_ROLE system role avoids having to grant full DBA authority to a user ID, thereby avoiding security problems associated with distributing DBA user IDs and passwords.

For example, a SQL Remote user ID with the SYS_RUN_REPLICATION_ROLE system role has no extra privileges on any connection apart from the Message Agent. Even if the user ID and password for this user is widely distributed, there is no security problem. As long as the user ID has no permissions beyond CONNECT granted on the database, no one can use this user ID to access data in the database.

Related Information

sp_has_role System Procedure
GRANT ROLE Statement

1.9.1.1.2 Compatibility Roles

Compatibility roles can be thought of as starter roles containing logical groups of privileges.

They are also present for backward compatibility with earlier versions of SQL Anywhere that support authority-based security.

You can still grant and revoke authorities using the deprecated GRANT and REVOKE syntax for doing so. However, the database server converts the grant or revoke into their compatibility role equivalent. Additionally, for the pre-16.0 authorities that were not inheritable (DBA, REMOTE DBA, BACKUP, RESOURCE and VALIDATE), their compatibility role equivalent is inheritable by default.

You cannot modify the underlying system privileges of compatibility roles. However, you can migrate them to user-defined roles, and then modify the privileges. When you migrate a compatibility role, all grantees of the compatibility role are automatically granted the user-defined role instead. Migrating a compatibility role to a user-defined role automatically drops the compatibility role. However, you can restore a compatibility role at any time.

Following is a table describing all compatibility roles, including the pre-16.0 version authority equivalent for each role.

Compatibility role	Description	Pre-16.0 authority
SYS_AUTH_DBA_ROLE	This role encompasses all grantable privileges in the software by virtue of being granted the SYS_AUTH_SA_ROLE and SYS_AUTH_SSO_ROLE compatibility roles.	DBA
SYS_AUTH_SA_ROLE	This role encompasses all **database administration** privileges that are found in SYS_AUTH_DBA_ROLE compatibility role, and is linked with that role. This role, plus the SYS_AUTH_SSO_ROLE, make up all of the privileges in the SYS_AUTH_DBA_ROLE compatibility role. This role can be migrated, dropped, and restored, but only as part of performing those operations on SYS_AUTH_DBA_ROLE.	-
SYS_AUTH_SSO_ROLE	This role encompasses all **security and access related privileges** that are found in the SYS_AUTH_DBA_ROLE compatibility role. This role, plus the SYS_AUTH_SA_ROLE, make up all of the privileges in the SYS_AUTH_DBA_ROLE compatibility role. This role can be migrated, dropped, and restored, but only as part of performing those operations on SYS_AUTH_DBA_ROLE.	-
SYS_AUTH_BACKUP_ROLE	This role allows a user to back up databases and transaction logs with archive or image backups by using the BACKUP DATABASE statement or dbbackup utility.	BACKUP
SYS_AUTH_PROFILE_ROLE	This role allows a user to perform profiling, tracing, and diagnostic operations.	PROFILE
SYS_AUTH_READCLIENTFILE_ROLE	This role allows a user to read files on the client computer, for example when loading data from a file on a client computer.	READCLIENTFILE

Compatibility role	Description	Pre-16.0 authority
SYS_AUTH_READFILE_ROLE	This role allows a user to use the OPENSTRING clause in a SELECT statement to read a file.	READFILE
SYS_AUTH_RESOURCE_ROLE	This role allows a user to create database objects, such as tables, views, stored procedures, and triggers.	RESOURCE
SYS_AUTH_VALIDATE_ROLE	This role allows a user to perform database, table, index, and checksum validation by using the VALIDATE statement or dbvalid utility.	VALIDATE
SYS_AUTH_WRITECLIENTFILE_ROLE	This role allows a user to write to files on a client computer, for example when using the UNLOAD TABLE statement to write data to a client computer.	WRITECLIENTFILE
SYS_AUTH_WRITEFILE_ROLE	This role allows a user to execute the xp_write_file system procedure.	WRITEFILE

In this section:

Related Information

Authorities Become Compatibility Roles [page 1659]
Changes in Inheritance Behavior for Some Authorities That Became Roles [page 1675]
GRANT ROLE Statement
REVOKE ROLE Statement

1.9.1.1.2.1 Migrating a Compatibility Role to a User-defined Role (SQL Central)

Migrate a compatibility role to a user-defined role, and then grant the new role to other roles and users.

Prerequisites

You must have the MANAGE ANY USER system privilege, the MANAGE ROLES system privilege, and administrative rights on the role being migrated.

Context

When you migrate a compatibility role, all grantees of the compatibility role are automatically granted the user-defined role, and the compatibility role is deleted.

Compatibility roles are like starter roles. You cannot modify the underlying system privileges of compatibility roles. However, you can migrate them to user-defined roles, and then modify the privileges.

The name you assign to the new role cannot begin with the prefix SYS_ and end with the suffix _ROLE. For example, SYS_MyBackup_ROLE cannot be the name of a user-defined role. The name you give the new role is permanent; you cannot alter it later.

The SYS_AUTH_SA_ROLE and SYS_AUTH_SSO_ROLE compatibility roles cannot be migrated individually. However, when the SYS_AUTH_DBA_ROLE compatibility role is migrated, SYS_AUTH_SA_ROLE and SYS_AUTH_SSO_ROLE are automatically migrated to new user-defined roles as well (sa_role and sso_role, by default).

If you migrate a role for which system privilege inheritance has been disabled (for example, SYS_AUTH_VALIDATE_ROLE), the privileges of the newly created user-defined role will be granted to all grantees in the inheritance tree.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, double-click *Roles* and select the compatibility role.
3. Right-click the compatibility role and then click *Migrate to User-defined Role*.
4. In the *Name* field, specify a new name for the new user-defined role, or accept the default name.
5. Click *OK*.

Results

All grantees of the compatibility role are automatically granted the user-defined role. The compatibility role is deleted.

Next Steps

Modify the system privileges of the user-defined role and grant the role to other users.

Related Information

Migrating a Compatibility Role to a User-defined Role (SQL) [page 1536]
Restoring a Compatibility Role (SQL) [page 1542]
Configuring Roles and Privileges For a Role (SQL Central) [page 1561]

1.9.1.1.2.2 Migrating a Compatibility Role to a User-defined Role (SQL)

Migrate a compatibility role to a user-defined role, and then grant the new role to other roles and users.

Prerequisites

You must have the MANAGE ROLES system privilege and administrative rights on the role being migrated.

Context

When you migrate a compatibility role, all grantees of the compatibility role are automatically granted the user-defined role, and the compatibility role is deleted.

Compatibility roles are like starter roles. You cannot modify the underlying system privileges of compatibility roles. However, you can migrate them to user-defined roles, and then modify the privileges.

The name you assign to the new role cannot begin with the prefix SYS_ and end with the suffix _ROLE. For example, SYS_MyBackup_ROLE cannot be the name of a user-defined role. The name you give the new role is permanent; you cannot alter it later.

If you migrate a role for which system privilege inheritance has been disabled (for example, SYS_AUTH_VALIDATE_ROLE), the privileges of the newly created user-defined role will be granted to all grantees in the inheritance tree.

The SYS_AUTH_SA_ROLE and SYS_AUTH_SSO_ROLE compatibility roles cannot be migrated individually. However, when the SYS_AUTH_DBA_ROLE compatibility role is migrated, SYS_AUTH_SA_ROLE and SYS_AUTH_SSO_ROLE are automatically included in the new user-defined role.

Procedure

To migrate a compatibility role, execute an ALTER ROLE statement similar to the following:

```
ALTER ROLE compatibility-role MIGRATE TO new-role-name;
```

If you are migrating the SYS_AUTH_DBA_ROLE compatibility role, remember to include names for the SYS_AUTH_SA_ROLE and SYS_AUTH_SSO_ROLE compatibility roles that get automatically migrated (that is, `MIGRATE TO new-role-name, new-sa-role-name, new-sso-role-name`).

Results

All grantees of the compatibility role are automatically granted the user-defined role. The compatibility role is deleted.

Example

The following statement migrates the SYS_AUTH_BACKUP_ROLE compatibility role to the new user-defined role, my_BACKUP. All users that were previously granted the SYS_AUTH_BACKUP_ROLE compatibility role are now granted the my_BACKUP role, including whatever administrative privileges they had over the compatibility role.

```
ALTER ROLE SYS_AUTH_BACKUP_ROLE MIGRATE TO my_BACKUP;
```

The following statement migrates the SYS_AUTH_DBA_ROLE, SYS_AUTH_SA_ROLE, and SYS_AUTH_SSO_ROLE compatibility roles to the user-defined roles my_DBA, my_DBA_Administration and my_DBA_Security, respectively. All users, underlying system privileges, and roles granted to the original roles are automatically migrated to the new roles. Finally, the SYS_AUTH_DBA_ROLE, SYS_AUTH_SA_ROLE, and SYS_AUTH_SSO_ROLE compatibility roles are dropped.

```
ALTER ROLE SYS_AUTH_DBA_ROLE
MIGRATE TO my_DBA, my_DBA_Administration, my_DBA_Security;
```

Next Steps

Modify the system privileges of the new user-defined role and grant the role to other users.

Although compatibility roles are dropped automatically after being migrated, you can restore them for future purposes using the CREATE ROLE statement.

Related Information

Migrating a Compatibility Role to a User-defined Role (SQL Central) [page 1535]
Restoring a Compatibility Role (SQL) [page 1542]
ALTER ROLE Statement

1.9.1.1.2.3 Granting a Compatibility Role (SQL Central)

Grant a compatibility role to a user or role.

Prerequisites

You must have administrative rights over the compatibility role you are granting.

Context

When you grant a compatibility role to a role, the compatibility role's system privileges are available to both the role and its grantees. However, you can disable the inheritance of the system privileges for the following compatibility roles:

- SYS_AUTH_DBA_ROLE
- SYS_AUTH_RESOURCE_ROLE
- SYS_AUTH_BACKUP_ROLE
- SYS_AUTH_VALIDATE_ROLE

In SQL Central, when you grant one of these compatibility roles to a user-extended role or a system role, the inheritance of the system privileges is disabled by default. Disabling system privilege inheritance for a compatibility role mimics the behavior of the non-inheritable authority in version 12 and earlier databases.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the *Roles* folder, click the compatibility role you want to grant.
3. In the right pane, click *Grantees*.
4. Right-click anywhere in the right pane and choose *New* > *Grantees*.
5. Click the user or role you want to grant the compatibility role to and then click *OK*.

   A new row is added in *Grantees*.

   If the inheritance of the system privileges for the compatibility role is disabled, then the *Options* column contains *Inheritance disabled*. To enable the inheritance of the system privileges, right-click the row, click *Set Options*, and choose *Enable system privilege inheritance*.

   With the exception of granting the SYS_AUTH_DBA_ROLE compatibility role, you cannot grant administrative rights if the *Inheritance disabled* is selected.
6. Click *File* > *Save*.

Results

The compatibility role is granted to the user or role.

Related Information

Authorities Become Compatibility Roles [page 1659]
Migrating a Compatibility Role to a User-defined Role (SQL Central) [page 1535]

1.9.1.1.2.4 Granting a Compatibility Role (SQL)

Grant a compatibility role to a user, or user-extended role.

Prerequisites

You must have administrative rights over the role you are granting.

Context

You cannot grant a compatibility role to a user-defined role that is not a user-extended role.

When you grant a compatibility role to a role, the compatibility role's system privileges are available to both the role and its grantees. However, you can disable the inheritance of the system privileges for the following compatibility roles:

- SYS_AUTH_DBA_ROLE
- SYS_AUTH_RESOURCE_ROLE
- SYS_AUTH_BACKUP_ROLE
- SYS_AUTH_VALIDATE_ROLE

When you disable the inheritance of a compatibility role's system privileges, the system privileges are available only to the role, not to its grantees. Disabling system privilege inheritance for a compatibility role mimics the behavior of the non-inheritable authority in version 12 and earlier databases.

Procedure

Execute a GRANT ROLE statement:

Option	Action
Grant the ability to exercise the role along with the ability to grant it to and revoke it from other users or roles:	`GRANT ROLE SYS_AUTH_DBA_ROLE TO user-or-role-name WITH ADMIN OPTION;`
Grant the ability to grant the role or revoke it from other users or roles, but not exercise the role:	`GRANT ROLE SYS_AUTH_DBA_ROLE TO user-or-role-name WITH ADMIN ONLY OPTION;`
Grant the ability to exercise the role but not grant it to, nor revoke it from, other users or roles:	`GRANT ROLE SYS_AUTH_DBA_ROLE TO user-or-role-name WITH NO ADMIN OPTION;`
Grant the ability to exercise a compatibility role and disable system privilege inheritance of the role:	`GRANT ROLE SYS_AUTH_DBA_ROLE TO role-name WITH NO SYSTEM PRIVILEGE INHERITANCE;`

Results

The compatibility role is granted to the user or role.

Example

This example grants the SYS_AUTH_BACKUP_ROLE to role R_HumanResources.

```
GRANT ROLE SYS_AUTH_BACKUP_ROLE TO R_HumanResources
WITH NO SYSTEM PRIVILEGE INHERITANCE;
```

Related Information

Authorities Become Compatibility Roles [page 1659]
Migrating a Compatibility Role to a User-defined Role (SQL) [page 1536]
GRANT ROLE Statement

1.9.1.1.2.5 Restoring a Compatibility Role (SQL Central)

Restore a compatibility role that has been migrated or dropped.

Prerequisites

You must have the MANAGE ROLES system privilege and have administrative rights on all of the system privileges granted to the role being restored. For example, to restore the SYS_AUTH_BACKUP_ROLE compatibility role, you must have the MANAGE ROLES system privilege, and administrative rights on the BACKUP DATABASE system privilege.

Context

You can restore compatibility roles that are no longer in the database. Restoring one of these roles can be helpful to remember the privileges the original role had or to migrate them to user-defined roles.

If you migrated the compatibility role (causing it to be dropped), grantees of the role were automatically granted the new user-defined role you migrated it to. Restoring a compatibility role does not restore the grantees back to the compatibility role. They remain grantees of the user-defined role you created.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, double-click *Roles*.

3. In the right pane, right-click anywhere in the pane and click *New* > *Roles*.
4. Click *Restore one or more compatibility roles* in the *Create Role Wizard* and follow the instructions.

Results

The role is restored. Administrative privileges on the restored role are automatically granted to the SYS_AUTH_DBA_ROLE compatibility role if it still exists in the database.

When you restore the SYS_AUTH_DBA_ROLE compatibility role, the SYS_AUTH_SA_ROLE and SYS_AUTH_SSO_ROLE compatibility roles are automatically restored as well.

Related Information

Migrating a Compatibility Role to a User-defined Role (SQL Central) [page 1535]
Dropping a Role (SQL Central) [page 1569]
CREATE ROLE Statement

1.9.1.1.2.6 Restoring a Compatibility Role (SQL)

Restore a compatibility role that has been migrated or dropped.

Prerequisites

You must have the MANAGE ROLES system privilege and have administrative rights on all of the system privileges granted to the role being restored. For example, to restore the SYS_AUTH_BACKUP_ROLE compatibility role, you must have the MANAGE ROLES system privilege, and administrative rights on the BACKUP DATABASE system privilege.

Context

You can restore compatibility roles that are no longer in the database. Restoring one of these roles can be helpful to remember the privileges the original role had or to migrate them to user-defined roles.

If you migrated the compatibility role (causing it to be dropped), grantees of the role were automatically granted the new user-defined role you migrated it to. Restoring a compatibility role does not restore the grantees back to the compatibility role. They remain grantees of the user-defined role you created.

Procedure

Execute a CREATE ROLE statement similar to the following, where `role-name` is the name of the role:

```
CREATE ROLE role-name;
```

Results

The role is restored. Administrative privileges on the restored role are automatically granted to the SYS_AUTH_DBA_ROLE compatibility role if it still exists in the database.

When you restore the SYS_AUTH_DBA_ROLE compatibility role, the SYS_AUTH_SA_ROLE and SYS_AUTH_SSO_ROLE compatibility roles are automatically restored as well.

Example

The following example restores the SYS_AUTH_BACKUP_ROLE compatibility role and grants user TomW the ability to grant the role or revoke it from other users or roles, but not exercise the role:

```
GRANT ROLE SYS_AUTH_BACKUP_ROLE TO TomW WITH ADMIN ONLY OPTION;
```

The following example restores the SYS_AUTH_DBA_ROLE, SYS_AUTH_SA_ROLE, and SYS_AUTH_SSO_ROLE compatibility roles:

```
CREATE ROLE SYS_AUTH_DBA_ROLE;
```

Related Information

Migrating a Compatibility Role to a User-defined Role (SQL) [page 1536]
CREATE ROLE Statement

1.9.1.1.3 User-defined Roles

A user-defined role is a collection you can create of system privileges, object-level privileges, and roles, typically created to group privileges related to a specific task or set of tasks.

You can create user-defined roles to suit the needs of your organization.

User-defined roles can own objects.

When a user-defined role is granted with administrative rights, a user can manage (grant, revoke, and drop) the role, as well as use any of the underlying system privileges of the role. When granted with administrative rights

only, a user can manage the role, but cannot use its underlying system privileges. When granted with no administrative rights, a user can only use its underlying system privileges.

One special form of user-defined role is the **user-extended role**. This is a user who has been extended to become a role, which can be granted to others.

In this section:

Related Information

User-extended Roles [page 1548]
Migrating a Compatibility Role to a User-defined Role (SQL) [page 1536]

1.9.1.1.3.1 COCKPIT_ROLE User-defined Role

The COCKPIT_ROLE is a user-defined role that is required to use the Cockpit.

Default

The COCKPIT_ROLE is a user-defined role that is created when you create or upgrade a database. By default, its administrative rights are granted to the MANAGE ROLES system role.

You must explicitly grant exercise rights for this role to your users. In newly created databases the exercise rights to the COCKPIT_ROLE are not granted to any user, including the initial user. In version 16 databases the COCKPIT_ROLE does not exist, so it must be created and then granted to users.

For convenience the COCKPIT_ROLE is populated with the following system privileges. A user that has the COCKPIT ROLE and these system privileges has full access to the features in Cockpit.

MONITOR system privilege
DROP CONNECTION system privilege
BACKUP DATABASE system privilege
SERVER OPERATOR system privilege
ACCESS DISK INFORMATION system privilege

Remarks

The COCKPIT_ROLE user-defined role is required to connect to the Cockpit. It is a database server level role.

If your database does not have the COCKPIT_ROLE and you need to use the Cockpit, then create the role and grant it to your users.

Because it is a user-defined role, you can delete this role from your database and recreate the role later if you need to. For example, to prevent all users from a database from being able to connect to the Cockpit, delete the COCKPIT_ROLE from the database.

You can grant or revoke any or all of the default privileges from the Cockpit. The effect of revoking a default privilege is that users can connect to the Cockpit, but any actions that require the revoked privilege are disabled. For example, suppose you have a user who only has the COCKPIT_ROLE. If you revoke the BACKUP DATABASE system privilege from the COCKPIT_ROLE, then this user cannot backup their database using the Cockpit or any other tool or utility.

The Cockpit runs on a database server, and it displays information about all databases running on that database server. Any database user with the COCKPIT_ROLE from any of the databases running on the database server can use the Cockpit. However to perform an action on a database, such as back up, you must provide credentials for a user of that database. For example, there are two databases running on the same server: database-A and database-B. You connect to the Cockpit using the credentials of a user from database-A who has the COCKPIT_ROLE including its default privileges. You have permission to back up database-A. But, to backup database-B, you must provide the credentials of a user of database-B, who has the required system privileges to back up a database.

Example

To recreate the COCKPIT_ROLE, execute the following SQL statement:

```
CREATE ROLE COCKPIT_ROLE;
GRANT MONITOR, DROP CONNECTION, BACKUP DATABASE, SERVER OPERATOR, ACCESS DISK
INFORMATION  TO COCKPIT_ROLE;
```

For example, execute the following statement to grant the COCKPIT_ROLE to the user JohnDoe:

```
GRANT ROLE COCKPIT_ROLE TO JohnDoe;
```

Related Information

SQL Anywhere Cockpit [page 1419]
Cockpit Security [page 1429]
Starting and Connecting to the Cockpit (sa_server_option System Procedure) [page 1423]
Starting and Connecting to the Cockpit(SQL Central) [page 1421]

1.9.1.1.3.2 Creating a User-defined Role (SQL Central)

Create a role, and grant roles and privileges to it.

Prerequisites

You must have the MANAGE ROLES system privilege.

The name you assign to the new role cannot begin with the prefix SYS_ and end with the suffix _ROLE. For example, SYS_MyBackup_ROLE cannot be the name of a user-defined role.

Context

You can also convert an existing user into a user-extended role, and then grant the user-extended role to other users.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, right-click *Roles* and click *New* > *Role*.
3. Follow the instructions in the wizard.

Results

A new role is created. By default, you are the owner of the role and the role is granted to the MANAGE ROLES system privilege.

Next Steps

Grant system privileges to this role and grant this role to other users.

Related Information

Roles [page 1527]

Role Administrators [page 1555]
Inheritance of Roles and Privileges [page 1617]
Granting a Role (SQL Central) [page 1563]
Converting a User to a User-extended Role (SQL Central) [page 1550]
CREATE ROLE Statement

1.9.1.1.3.3 Creating a User-defined Role (SQL)

Create a role and grant privileges and roles to it.

Prerequisites

You must have the MANAGE ROLES system privilege.

The name you assign to the new role cannot begin with the prefix SYS_ and end with the suffix _ROLE. For example, SYS_MyBackup_ROLE cannot be the name of a user-defined role.

Context

You can also convert an existing user into a user-extended role, and then grant the user-extended role to other users.

Procedure

1. Connect to the database.
2. Execute a CREATE ROLE statement. For example:

   ```
   CREATE ROLE role-name
   ```
3. Grant system privileges and roles to the new role.

Results

The new role is created.

Example

The following statement creates a role called Sales. Because no administrator was specified, global administrators (users with the MANAGE ROLES system privilege) can administrator the role.

```
CREATE ROLE Sales;
GRANT SELECT, UPDATE ON GROUPO.SalesOrders TO Sales;
GRANT SELECT, UPDATE ON GROUPO.SalesOrderItems TO Sales;
```

Next Steps

Grant system privileges to this role and grant this role to other users.

Related Information

Roles [page 1527]
Role Administrators [page 1555]
Inheritance of Roles and Privileges [page 1617]
Creating a User-defined Role (SQL Central) [page 1546]
Converting a User to a User-extended Role (SQL Central) [page 1550]
Granting a Role (SQL Central) [page 1563]
CREATE ROLE Statement
GRANT ROLE Statement

1.9.1.1.4 User-extended Roles

A **user-extended role** is a user that has been extended to be a role, and is a type of user-defined role.

It is a user ID that has been extended to act as a role that can be granted to others. User-extended roles are the equivalent of **groups** in pre-16.0 releases of SQL Anywhere.

When you grant a user-extended role to a user or another role, the grantee inherits all the system and object-level privileges that the user-extended role has, including any administration rights.

Because ownership of database objects is associated with a single user ID, when the owner is a user-extended role, ownership of the database object is not inherited by its grantees. Only granted privileges are inherited.

User-extended roles are convenient when you have a user with a set of system privileges that you want to grant to another user. The user who has become a user-extended role can administer the new role (grant and revoke it to others) unless this privilege is explicitly removed.

When you create a user-extended role, the MANAGE ROLES system privilege is automatically granted the role with administrative rights only.

You can revoke the extension of a role, and control what privileges are revoked from the grantees.

A user can still log in using their user ID even when they are a user-extended role.

To convert a user to a user-extended role, use the CREATE ROLE statement. To convert a user-extended role back to a user, use the DROP ROLE statement.

Example

- The following statements creates user1; grants the BACKUP DATABASE privilege to user1; extends user1 as a role; and then grants the user1 role to user2 and user3:

  ```
  CREATE USER "user1" IDENTIFIED BY 'sql';
  GRANT BACKUP DATABASE TO "user1";
  CREATE ROLE FOR USER "user1";
  GRANT ROLE "user1" TO "user2";
  GRANT ROLE "user1" TO "user3";
  ```

  The following statement revokes user1 role from user2:

  ```
  REVOKE ROLE "user1" FROM "user2";
  ```

  The following statement changes user1 role to a user. As a consequence, the user1 role is revoked from user3:

  ```
  DROP ROLE FROM USER "user1" WITH REVOKE;
  ```

In this section:

Converting a User to a User-extended Role (SQL Central) [page 1550]
Change a user to a user-extended role, which can then be granted to other users and roles.

Converting a User to a User-extended Role (SQL) [page 1551]
Change a user to a user-extended role, which can then be granted to other users and roles.

Converting a User-extended Role Back to a User (SQL Central) [page 1553]
Change a user-extended role back to a regular user.

Converting a User-extended Role Back to a User (SQL) [page 1554]
Change a user-extended role back to a regular user.

Related Information

Database Object Names and Prefixes
sp_has_role System Procedure
CREATE ROLE Statement
DROP ROLE Statement
GRANT ROLE Statement
REVOKE ROLE Statement

1.9.1.1.4.1 Converting a User to a User-extended Role (SQL Central)

Change a user to a user-extended role, which can then be granted to other users and roles.

Prerequisites

You must have the MANAGE ROLES privilege.

Context

User-extended roles are convenient when you have a user with a desired set of system privileges and roles that you also want to grant to another user.

The user who has become a user-extended role can administer the new role (grant and revoke it to others) unless this privilege is explicitly removed. Also, users with the MANAGE ROLES system privileges can administer the role.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, double-click *Users* and select the user.
3. Right-click the user and then click *Change to User-extended Role*.

Results

The user is extended into a role. A *Grantees* tab appears in the right pane, and new role appears in the list of roles in the left pane.

Next Steps

You can grant the new role to other users and roles. When you grant a user-extended role to a user or another role, the grantee inherits all the system and object-level privileges that the user-extended role has, including any administrative rights.

Related Information

Converting a User-extended Role Back to a User (SQL Central) [page 1553]
Converting a User to a User-extended Role (SQL) [page 1551]
CREATE ROLE Statement

1.9.1.1.4.2 Converting a User to a User-extended Role (SQL)

Change a user to a user-extended role, which can then be granted to other users and roles.

Prerequisites

You must have the MANAGE ROLES system privilege.

Context

User-extended roles are convenient when you have a user with a desired set of system privileges and roles that you also want to grant to another user.

You can specify administrators for the role during the conversion. If you do not specify administrators, by default any user with MANAGE ROLE system privilege can administer the role.

The user being extended to be a user-extended role cannot be specified as an administrator, nor be granted administrative rights over the role after the conversion is complete. However, if the user being extended has the MANAGE ROLES system privilege, and no administrators are specified at conversion time, then they can administer the role.

Procedure

1. Connect to the database.
2. Execute a CREATE ROLE statement similar to the following:

Option	Action
Extend the user to be a role that any user with the MANAGE ROLES system privilege can administer.	`CREATE ROLE FOR USER userid;`
Extend the user to be a role that any user in `list-of-administrator-ids` can administer.	`CREATE ROLE FOR USER userid`

Option	Action
	`WITH ADMIN list-of-administrator-ids;`

Results

The user is extended into a role.

Example

The following statement extends user JaneSmith to become a role that can be assigned to others.

```
CREATE ROLE FOR USER JaneSmith;
```

Next Steps

You can grant the new role to other users and roles. When you grant a user-extended role to a user or another role, the grantee inherits all the system and object-level privileges that the user-extended role has, including any administrative rights.

Related Information

Converting a User-extended Role Back to a User (SQL) [page 1554]
Converting a User to a User-extended Role (SQL Central) [page 1550]
CREATE ROLE Statement

1.9.1.1.4.3 Converting a User-extended Role Back to a User (SQL Central)

Change a user-extended role back to a regular user.

Prerequisites

You must have the MANAGE ROLES system privilege, or have administrative rights on the role.

Context

A user-extended role can be converted back to a regular user as long as all dependent roles meet the minimum required number of administrative users with active passwords, as set by the min_role_admin database option.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, double-click *Roles* and select the user-extended role.
3. Right-click the user-extended role and then click *Change to User*.

Results

The role is converted back to a user. The *Grantees* tab disappears in the right pane, and the role disappears from the list of roles in the left pane. Any objects that were owned by the user-extended role remain with the converted user. Any users or roles that previously had the user-extended role no longer have the privileges that the converted user has.

Next Steps

If you are changing the user-extended role to a user to delete the role, you can delete the user.

Related Information

DROP ROLE Statement

min_role_admins Option [page 773]

1.9.1.1.4.4 Converting a User-extended Role Back to a User (SQL)

Change a user-extended role back to a regular user.

Prerequisites

You must have the MANAGE ROLES system privilege, or have administrative rights on the role.

Context

A user-extended role can be converted back to a regular user as long as all dependent roles meet the minimum required number of administrative users with active passwords, as set by the min_role_admin database option.

Procedure

1. Connect to the database.
2. Execute a statement similar to the following:

Option	Statement
Convert the role back to a user, if the role is not granted to any roles or users	`DROP ROLE FROM USER userid;`
Convert the role back to a user, and revoke the underlying privileges from anyone the user-extended role had been granted to	`DROP ROLE FROM USER userid WITH REVOKE;`

Results

The role is converted back to a user. Any objects that were owned by the user-extended role remain with the converted user. Any users or roles that previously had the user-extended role no longer have the privileges that the converted user has.

Example

The following statement converts a user-extended role named Joe back to a regular user. Objects owned by the user-extended role are now owned by the regular user, Joe. Users or roles that had been granted Joe retain the underlying privileges associated with the role.

```
DROP ROLE FROM USER Joe;
```

The following statement converts a user-extended role named Sam back to a regular user. Users and roles who had been granted Sam will have the privileges of Jack revoked.

```
DROP ROLE FROM USER Sam WITH REVOKE;
```

The following statement drops a role named Marketing2, drops the objects it owned, and revokes its underlying system privileges from those who had been granted the role.

```
DROP ROLE Marketing2 WITH REVOKE WITH DROP OBJECTS;
```

Next Steps

If you are changing the user-extended role to a user to delete the user, you can now delete the user.

Related Information

Converting a User-extended Role Back to a User (SQL Central) [page 1553]
Converting a User to a User-extended Role (SQL) [page 1551]
DROP ROLE Statement
min_role_admins Option [page 773]

1.9.1.1.5 Role Administrators

Role administrators are responsible for granting and revoking user-defined roles to users and other roles.

You can add and remove role administrators as needed. There is no maximum number of role administrators that can be granted to a single role. However, there is a minimum number, as specified by the min_role_admins database option.

There are also **global role administrators**, which are any users that have the MANAGE ROLES system privilege. Global role administrators can administer user-defined roles for which no administrator has been specified.

Global role administrators are the sole administrators of the system roles.

Specify Administrator

You can specify role administrators at role creation time, or after the role has been created.

In this example, the role myRole is created, and MaryM and TomS are granted the role with administrative rights. After creation, only MaryM and TomS will be able to grant or revoke the role.

```
CREATE ROLE myRole WITH ADMIN MaryM, TomS;
```

In this example, the role myRole is created, a privilege is added to it, and then the user ID PeriW is added as an administrator (only).

```
CREATE USER PeriW IDENTIFIED BY PeriW;
CREATE ROLE myRole;
GRANT SELECT ANY TABLE to myRole;
GRANT ROLE myRole to PeriW WITH ADMIN ONLY OPTION;
```

Global Administrators and the MANAGE ROLES System Privilege

When you create a role but do not specify an administrator during the creation, the MANAGE ROLES system privilege is automatically granted administrative rights (only) to the new role. This way, any user with the MANAGE ROLES system privilege can administer the role.

MANAGE ROLES is unique in this way: it is the only system privilege that can have roles granted to it (WITH ADMIN ONLY OPTION) and revoked from it.

If no administrator is specified at role creation time (causing it to be automatically granted to MANAGE ROLES) and then administrators are specified for the roles later, then those administrators, plus anyone with the MANAGE ROLES system privilege, can administer the role.

If you create a role and specify administrators during the creation, only those administrators can administer the role. Global administrators cannot administer the role unless MANAGE ROLES was one of the administrators you specified. To create a role and include MANAGE ROLES as one of the administrators, you execute a statement similar to the following. When specifying the MANAGE ROLES system privilege, you must use its internal representation, SYS_MANAGE_ROLES_ROLE.

When you grant a role to the MANAGE ROLES system privilege, you must grant it with administrative rights only (WITH ADMIN ONLY); otherwise, an error is returned.

```
CREATE ROLE myRole WITH ADMIN ONLY MaryM, TomS, SYS_MANAGE_ROLES_ROLE;
```

MaryM, TomS, and any user with the MANAGE ROLES system privilege can administer the role. However, MaryM and TomS will not be able to exercise the role.

> **i Note**
>
> When creating a role, the administration-related options are: WITH ADMIN, and WITH ADMIN ONLY. When granting a role, the administration-related options are: WITH ADMIN OPTION, WITH NO ADMIN OPTION, and WITH ADMIN ONLY OPTION.

You can also grant administration rights to the MANAGE ROLES system privilege after a role is created, by executing a statement similar to the following. Again, you must remember to specify WITH ADMIN ONLY OPTION:

```
GRANT ROLE myRole TO SYS_MANAGE_ROLES_ROLE WITH ADMIN ONLY OPTION;
```

If you created a role and you want to remove the global administrator's ability to administer, you can revoke MANAGE ROLES from it by executing a statement similar to the following:

```
REVOKE ROLE myRole FROM SYS_MANAGE_ROLES_ROLE;
```

Changing or Dropping Role Administrators After Creation

If you have a role that already has an administrator assigned to it and you want to change it to another user ID, grant the role to the new administrator (specifying WITH ADMIN or WITH ADMIN ONLY), and then revoke administrator privileges from the first administrator.

> **i** Note
>
> If you only want to add role administrators to a role, use the GRANT ROLE statement with the WITH ADMIN [ONLY] OPTION clause instead of the CREATE OR REPLACE ROLE statement.

The min_role_admins database option controls the minimum number of administrators a role should have. If you try to revoke administration rights from a user, the statement can fail if doing so would mean there are fewer administrators than the value set by the min_role_admins option. However, this is not a problem as long as you have other users who have global role administration rights (MANAGE ROLE system privilege), as these global role administrators can administer roles for which no administrators have been specified.

Specify an Administrator During Role Creation

When you create a role, set the administrators as part of the CREATE ROLE statement. If you do so, consider granting the MANAGE ROLES system privilege to the administrators. For example, the following statement creates the role myRole and sets SallyG and IrisM as administrators:

```
CREATE ROLE myRole WITH ADMIN SallyG, IrisM, SYS_MANAGE_ROLES_ROLE;
```

In this section:

Super-users [page 1558]

Super-users are users that can exercise any system privilege and administer any role; they can perform any privileged operation in a database.

Related Information

CREATE ROLE Statement
GRANT ROLE Statement
min_role_admins Option [page 773]

1.9.1.1.5.1 Super-users

Super-users are users that can exercise any system privilege and administer any role; they can perform any privileged operation in a database.

When a new database is created, the initial user created is the **database super-user**, with full administrative and exercise rights on all system privileges except the ability to reset passwords. This user can perform any privileged operation in a newly created database. This user is often referred to as the DBA user.

A **database server super-user** is a database super-user that also has all exercise rights on the COCKPIT_ROLE user-defined role.

If you create a new role and assign administrators as part of role creation, administration is then limited to the grantees who were given administration rights. Therefore, if you want your database super-user to have administrative rights for the new role, then you must grant that to the user. Additionally, you can delete the initial user and create your own super-user.

To create a database super-user, create a user and grant the SYS_AUTH_SA_ROLE and SYS_AUTH_SSO_ROLE system roles to the user. By doing so, you grant them all possible privileges in the system, including any new privileges that are introduced in upgrades.

Related Information

Compatibility Roles [page 1532]
GRANT ROLE Statement

1.9.1.1.6 System Privileges Introduced in Upgrades

A new release of the software may introduce new system privileges.

In a *new database*, these privileges are automatically included in either the SYS_AUTH_SA_ROLE or SYS_AUTH_SSO_ROLE compatibility roles, depending on the type of privileged operation they allow.

In an *upgraded database*, these privileges are added to the UPGRADE ROLE system privilege (only). This behavior requires you to decide which roles and users you want to grant the new privileges to. After you have granted the new privileges, you should revoke them from the UPGRADE ROLE system privilege.

> **i** Note
>
> Even though UPGRADE ROLE is a system privilege, it is similar to a role because it can have privileges granted to it.

In this section:

1.9.1.1.6.1 Distributing Privileges Granted to the UPGRADE ROLE System Privilege After an Upgrade

Grant the privileges that have been added to the UPGRADE ROLE system privilege to other users or roles (except for compatibility roles), and then revoke the privileges from UPGRADE ROLE.

Prerequisites

You must have exercise and administration rights for the UPGRADE ROLE system privilege.

Context

Perform this procedure after upgrading when new privileges have been added to the UPGRADE ROLE system privilege. To determine if new privileges have been added for the release, view the UPGRADE ROLE in SQL Central. This is the same process as viewing the roles and privileges for a user or role. You can also look for mentions of newly added privileges in the list of new features for the release.

If you simply upgrade a database without following the steps described here, the database may encounter errors during a subsequent rebuild.

Procedure

1. In Interactive SQL, log in as a user with administration and exercise rights on the UPGRADE ROLE system privilege, and execute a statement that calls the sp_displayroles system procedure to display the privileges that have been granted to the UPGRADE ROLE system privilege. Note that when executing this statement, you must use the internal representation of the UPGRADE ROLE system privilege, which is SYS_UPGRADE_ROLE_ROLE:

   ```
   CALL sp_displayroles ( 'SYS_UPGRADE_ROLE_ROLE', 'expand_down' );
   ```

2. Grant the privileges listed to other users or roles, ensuring that you grant administration rights to at least one other user or role. You cannot assign the privilege to a compatibility role.
3. For each privilege you granted, log in as a user with administration rights for that privilege, and execute a REVOKE statement to revoke the privilege from the UPGRADE ROLE system privilege (again, use the internal representation, SYS_UPGRADE_ROLE_ROLE). For example:

   ```
   REVOKE new-privilege-name FROM SYS_UPGRADE_ROLE_ROLE;
   ```

Results

All new privileges have been granted to other users or roles, and the UPGRADE ROLE has no privileges granted to it.

Example

Here is an example for the OFFLINE RESET PASSWORD system privilege.

```
CALL sp_displayroles ( 'SYS_UPGRADE_ROLE_ROLE', 'expand_down' );
GRANT OFFLINE RESET PASSWORD TO DBA WITH ADMIN OPTION;
REVOKE OFFLINE RESET PASSWORD FROM SYS_UPGRADE_ROLE_ROLE;
```

Next Steps

None.

Related Information

Viewing the Roles and Privileges for a User or Role (SQL) [page 1613]
Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
Granting a System Privilege (SQL Central) [page 1598]
Granting a System Privilege (SQL) [page 1599]
Revoking a System Privilege (SQL Central) [page 1603]
Revoking a System Privilege (SQL) [page 1605]

1.9.1.1.7 Configuring Roles and Privileges For a Role (SQL Central)

Grant roles or privileges to any role, revoke roles or privileges from it, and set the administrative rights the role has over the roles and privileges it has been granted.

Prerequisites

If you are granting or revoking a system role you must have the MANAGE ROLES system privilege.

If you are granting or revoking a compatibility role, you must have administration rights for the role.

If you are granting or revoking SYS_RUN_REPLICATION_ROLE, you must have the SYS_REPLICATION_ADMIN_ROLE system role.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, double-click *Roles* and select the role to alter.

Option	Action
Grant a privilege	1. In the right pane, click the relevant *Privileges* tab. 2. Right-click anywhere in the right pane and click *New* > *Privileges*. 3. Choose one or more privileges from the list and click *OK*.
Revoke a privilege	1. In the right pane, click the relevant *Privileges* tab. 2. Right-click the privilege you want to revoke and click *Delete*.
Grant a role	1. In the right pane, click the *Roles* tab. 2. Right-click anywhere in the right pane and click *New* > *Granted Roles*. 3. Choose one ore more roles to grant and click *OK*.
Revoke a role	1. In the right pane, click the *Roles* tab. 2. Right-click the role you want to revoke and click *Delete*.

3. Click *File* > *Save*.

Results

The role is configured.

Related Information

Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
GRANT Statement
REVOKE Statement
REVOKE ROLE Statement

1.9.1.1.8 Configuring Roles and Privileges For a Role (SQL)

Grant roles or privileges to any role, revoke roles or privileges from it, and set the administrative rights the role has over the roles and privileges it has been granted.

Prerequisites

If you are granting or revoking a system role you must have the MANAGE ROLES system privilege.

If you are granting or revoking a compatibility role, you must have administration rights for the role.

If you are granting or revoking SYS_RUN_REPLICATION_ROLE, you must have the SYS_REPLICATION_ADMIN_ROLE system role.

Procedure

1. Connect to the database.
2. Execute statements similar to the following, according to the change you want to make:

Option	Action
Grant a privilege	`GRANT privilege-name TO target-role-name;`
Revoke a privilege	`REVOKE privilege-name FROM target-role-name;`
Grant a role	`GRANT ROLE role-name TO target-role-name;`
Revoke a role	`REVOKE ROLE role-name FROM target-role-name;`

Results

The role is configured.

Example

To grant the CREATE ANY OBJECT system privilege to the role RoleA without giving RoleA administrative rights, execute the following statement:

```
GRANT CREATE ANY OBJECT TO RoleA;
```

To grant RoleA the CREATE ANY OBJECT system privilege along with the ability to grant or revoke the system privilege to and from other users and roles, execute the following statement:

```
GRANT CREATE ANY OBJECT TO RoleA WITH ADMIN OPTION;
```

To grant RoleB along with its administrative rights to user Jane, execute the following statement:

```
GRANT ROLE RoleB TO Jane WITH ADMIN OPTION;
```

To grant user John the administrative rights to RoleB, but the inability to use RoleB, execute the following statement:

```
GRANT ROLE RoleB TO John WITH ADMIN ONLY OPTION;
```

Related Information

Configuring Roles and Privileges For a Role (SQL Central) [page 1561]
Viewing the Roles and Privileges for a User or Role (SQL) [page 1613]
GRANT Statement
REVOKE Statement
GRANT ROLE Statement
REVOKE ROLE Statement

1.9.1.1.9 Granting a Role (SQL Central)

Grant a role to a user or another role.

Prerequisites

You must have administrative rights on the role you are granting.

Context

You cannot grant administrative rights on a system role; only users with the MANAGE ROLES system privilege have administrative rights for system roles.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. Double-click *Users* or *Roles* and select a user or role that you want to grant the role(s) to.
3. In the right pane, click *Grantees*.
4. Right-click anywhere in the right pane and choose *New* > *Grantees*.
5. Click the user or role you want to grant the role to and then click *OK*.
6. (Optional) Use the *Adm.* (administrative rights) column to change the administrative rights for the role. An empty column indicates that the user or role does not have the administrative rights for the role. A checkmark in the column means the user or role can administer (grant or revoke) the role. You cannot grant administrative rights on a system role; only users with the MANAGE ROLES system privilege can administer system roles. Click in the column to toggle the option.

   (Optional) Use the *Exe.* (exercise rights) column to change the exercise rights for the user or role. A checkmark in the column means the user or role can exercise the role. The exercise privilege is granted by default. Click in the column to toggle the option.
7. Click *File* > *Save*.

Results

The role is granted to the user or role.

Related Information

Inheritance of Roles and Privileges [page 1617]
Changes to the GRANT Statement Syntax [page 1664]
Granting an Object-level Privilege (SQL Central) [page 1601]
Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
GRANT ROLE Statement

1.9.1.1.10 Granting a Role (SQL)

Grant a role to a user or another role.

Prerequisites

You must have administrative rights on the role you are granting.

Context

You cannot grant administrative rights on a system role; only users with the MANAGE ROLES system privilege can administer system roles.

Procedure

1. Connect to the database.
2. Execute a GRANT ROLE statement. For example:

Option	Action
Grant a role with no administrative rights.	`GRANT ROLE role-name TO userid;`
Grant a role with administrative rights.	`GRANT ROLE role-name TO userid WITH ADMIN OPTION;`
Grant administrative rights only on a role.	`GRANT ROLE role-name TO userid WITH ADMIN ONLY OPTION;`

Results

The role is granted to the user or role.

Example

To grant RoleB along with its administrative rights to the user Jane, execute the following statement:

```
GRANT ROLE RoleB TO Jane WITH ADMIN OPTION
```

To grant the user John the administrative rights to RoleB, but the inability to use RoleB, execute the following statement:

```
GRANT ROLE RoleB TO John WITH ADMIN ONLY OPTION
```

Related Information

Inheritance of Roles and Privileges [page 1617]
Changes to the GRANT Statement Syntax [page 1664]
Granting an Object-level Privilege (SQL Central) [page 1601]
Viewing the Roles and Privileges for a User or Role (SQL) [page 1613]
GRANT ROLE Statement

1.9.1.1.11 Revoking a Role (SQL Central)

Revoke a role from a user, a user-extended role, or a user-defined role.

Prerequisites

You must have administrative rights for the role to revoke user-defined roles. You must have MANAGE ROLES system privilege to revoke system roles.

You cannot revoke roles from the SYS role. You can revoke roles from other system roles, provided that these roles are not the default roles for the system role.

You can only revoke a role if all dependent roles meet the minimum required number of administrative users with active passwords, as set by the min_role_admins database option.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click either *Users* or *Roles* and then click a specific user or role.
3. Choose one of the following options:

Administrative option	Action
Revoke the role from a user or role that has been granted this role.	1. Click the *Grantees* tab. 2. To revoke the role from a user or role, right-click the user or role and click *Delete*.
Revoke exercise or administrative rights for a role.	1. Click the *Roles* tab. 2. Select a role, and then right-click and choose: • *Revoke* > *Exercise rights*. This is equivalent to executing the following statement: `REVOKE ROLE "role-name" FROM "user/role-name"` • *Revoke* > *Administrative rights*. This is equivalent to executing the following statement: `REVOKE ADMIN OPTION FOR ROLE "role-name" FROM "user/role-name"`

→ Tip

To undo your changes, select one or more rows and click *Edit* > *Undo*.

To turn the *Legend* on or off, click *File* > *Show Legend*.

4. Click *File* > *Save*.

Results

The role is revoked from the specified user or role.

Related Information

Migrating a Compatibility Role to a User-defined Role (SQL Central) [page 1535]
REVOKE ROLE Statement

1.9.1.1.12 Revoking a Role (SQL)

Revoke a role from a user, a user-extended role, or a user-defined role.

Prerequisites

You must have administrative rights for the role to revoke user-defined roles. You must have MANAGE ROLES system privilege to revoke system roles.

You cannot revoke roles from the SYS role. You can revoke roles from other system roles, provided that these roles are not the default roles for the system role.

You can only revoke a role if all dependent roles meet the minimum required number of administrative users with active passwords, as set by the min_role_admins database option.

Procedure

1. Connect to the database.
2. Execute a REVOKE ROLE statement. For example:

Option	Action
Revoke a role	`REVOKE ROLE role-name FROM userid;`
Revoke administrative rights for a role, but user can still exercise the role	`REVOKE ADMIN OPTION FOR ROLE role-name FROM userid;`

Results

The role is revoked from the specified user or role.

Example

Revoke the SYS_AUTH_RESOURCE_ROLE compatibility role from user Jim.

```
REVOKE ROLE SYS_AUTH_RESOURCE_ROLE FROM Jim;
```

Revoke only the administrative rights on the myRole role from user AnnW.

```
REVOKE ADMIN OPTION FOR ROLE myRole FROM AnnW;
```

Related Information

Revoking a Role (SQL Central) [page 1566]
Migrating a Compatibility Role to a User-defined Role (SQL Central) [page 1535]
REVOKE ROLE Statement
GRANT ROLE Statement

1.9.1.1.13 Dropping a Role (SQL Central)

Drop a user-defined, user-extended, or compatibility role.

Prerequisites

You must have administrative rights for the role being dropped.

The role being dropped must not own any database objects.

You can only drop a role if all dependent roles meet the minimum required number of administrative users with active passwords, as set by the min_role_admins database option.

Context

You cannot drop system roles.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. Double-click *Roles* and select the role you want to drop.
3. Right-click the role and click *Delete*.

Results

The role is dropped and revoked from any user or role that had been granted the role. When you drop a role that can be administered by the MANAGE ROLES system privilege, the role is also revoked from the MANAGE ROLES system privilege.

Related Information

Converting a User-extended Role Back to a User (SQL Central) [page 1553]
DROP ROLE Statement
REVOKE ROLE Statement
min_role_admins Option [page 773]

1.9.1.1.14 Dropping a Role (SQL)

Drop a user-defined, user-extended, or compatibility role.

Prerequisites

You must have administrative rights for the role being dropped.

You can only drop a role if all dependent roles meet the minimum required number of administrative users with active passwords, as set by the min_role_admins database option.

Context

You cannot drop system roles.

Procedure

Execute a DROP ROLE statement. For example:

```
DROP ROLE role-name;
```

If the role is assigned to users, you must specify the WITH REVOKE clause to drop the role:

```
DROP ROLE role-name WITH REVOKE;
```

If the role owns objects, you must specify the WITH DROP OBJECTS clause to drop the role:

```
DROP ROLE role-name WITH DROP OBJECTS;
```

Results

The role is dropped.

Example

Suppose no-one in your organization has been granted the SYS_AUTH_VALIDATE_ROLE compatibility role. You can drop it by executing the following statement:

```
DROP ROLE SYS_AUTH_VALIDATE_ROLE;
```

If there are users who have been granted the SYS_AUTH_VALIDATE_ROLE compatibility role, you can still drop it by executing the following statement:

```
DROP ROLE SYS_AUTH_VALIDATE_ROLE WITH REVOKE;
```

The following statement drops the SYS_AUTH_DBA_ROLE system role. This operation is possible if all of the system privileges it comes with are accounted for in other roles, with the appropriate number of administrators for those roles, as specified by the min_role_admins database option. The WITH REVOKE clause is added because the role has likely been assigned to other users.

```
DROP ROLE SYS_AUTH_DBA_ROLE WITH REVOKE;
```

Related Information

Compatibility Roles [page 1532]
min_role_admins Option [page 773]
DROP ROLE Statement

1.9.1.2 Privileges

A privilege is a right to perform a privileged operation on the system.

For example, altering a table is a privileged operation, depending on the type of alteration you are making. There are two types of privileges: system privileges and object-level privileges. **System privileges** give you the general right to perform a privileged operation, while **object-level privileges** restrict you to performing the operation on a specific object. For example, if you have the ALTER ANY TABLE system privilege, you can alter

any table in the system. If you do not, you can only edit tables you create or tables on which you have the ALTER TABLE object-level privilege.

System privileges are built in to the database and can be granted or revoked, but not created or dropped. With the exception of the MANAGE ROLES and UPGRADE ROLE privileges, system privileges cannot have system privileges granted to, or revoked from, them. Each system privilege, with the exception of the SET USER system privilege, is granted by default to either the SYS_AUTH_SA_ROLE or SYS_AUTH_SSO_ROLE compatibility role, but not both. The SET USER system privilege is granted to both roles (WITH ADMIN OPTION to SYS_AUTH_SSO_ROLE and WITH NO ADMIN OPTION to SYS_AUTH_SA_ROLE).

You grant and revoke system and object-level privileges by using the GRANT and REVOKE statements.

In this section:

Related Information

Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
Viewing the Roles and Privileges for a User or Role (SQL) [page 1613]

sp_has_role System Procedure
GRANT Statement
REVOKE Statement

1.9.1.2.1 Object-level Privileges

There are several object-level privileges that can be granted.

Privilege name	Supported by database objects	Inherited through group membership?	WITH GRANT OPTION allowed when granting?	Description
ALL	Tables, views	Yes	Yes	Allows a user to perform all tasks associated with the database object. This privilege grants the following privileges on tables: ALTER, DELETE, INSERT, REFERENCES, SELECT, and UPDATE. This privilege grants the following privileges on views: DELETE, INSERT, SELECT, and UPDATE. Granting ALL also grants LOAD and TRUNCATE privileges.
ALTER	Tables	Yes	Yes	Allows a user to alter the structure of a table or create a trigger on the table. Because this privilege grants the user the privilege to modify the database schema, it should not be granted to most users.
CONNECT	Databases	No	No	Allows a user to connect to the current database.
CONSOLIDATE	Users	No	No	Allows a user to identify a consolidated database in SQL Remote.

Privilege name	Supported by database objects	Inherited through group membership?	WITH GRANT OPTION allowed when granting?	Description
CREATE ON	Dbspaces	Yes	Yes	Allows a user to create on the dbspace. The additional privileges required depend on the object that is being created. For example, to create a table, one of CREATE TABLE, CREATE ANY TABLE, or CREATE ANY OBJECT is required.
DELETE	Tables, views	No	Yes	Allows a user to delete rows from the table or view.
EXECUTE	Procedures, user-defined functions	No	No	Allows a user to execute the procedure or user-defined function.
INSERT	Tables, views	Yes	Yes	Allows a user to insert rows into the table or view.
INTEGRATED LOGIN	Databases	No	No	Allows a user to connect to the current database using an integrated login.
KERBEROS LOGIN	Databases	No	No	Allows a user to connect to the current database using a Kerberos login.
LOAD	Tables	Yes	Yes	Allows a user to load the table if the -gl database option is set to anything other than NONE.
PUBLISH	Users	No	No	Identifies the user ID for a database in SQL Remote.

Privilege name	Supported by database objects	Inherited through group membership?	WITH GRANT OPTION allowed when granting?	Description
REFERENCES	Tables	Yes	Yes	Allows a user to create indexes on the table and to create foreign keys that reference the table. This privilege can also be granted on individual columns in a table. Because this privilege allows the user to modify the database schema, it should not be granted to most users. When the column names are specified, the user is only allowed to index those columns in the table.
REMOTE	Users	No	No	Allows a user to identify a remote database in SQL Remote and MobiLink.
SELECT	Tables, views	Yes	Yes	Allows a user to query the table or a view. This privilege can also be granted to individual columns in a table. When the column names are specified, the user is only allowed to view those columns in the table.
TRUNCATE	Tables, materialized views	Yes	Yes	Allows a user to truncate the table or materialized view.
UPDATE	Tables, views	Yes	Yes	Allows a user to update rows in the table or view. This privilege can also be granted to individual columns in a table.

Privilege name	Supported by database objects	Inherited through group membership?	WITH GRANT OPTION allowed when granting?	Description
USAGE	Sequence generators	No	Yes	Allows a user to evaluate the current or next value in the sequence.

Additional Notes

The WITH GRANT OPTION is not inherited by group members

While some object-level privileges are inheritable, the WITH GRANT OPTION is never inheritable. For example, if a user-extended role, UserA, is granted SELECT on a table and the WITH GRANT OPTION is specified, then UserA can select on the table, and grant select on that table to others. Grantees of that user-extended role inherit the ability to select from the table, but do not inherit the ability to grant the SELECT privilege to others.

Views

You can grant privileges on disabled views. Privileges for disabled views are stored in the database and become effective when the view is enabled.

Procedures

A user must own a procedure, or have EXECUTE privilege on it, to execute it.

Additionally, if the procedure definition includes SQL SECURITY DEFINER, the procedure runs with the privileges of the procedure owner, not the user calling the procedure. If the procedure definition includes SQL SECURITY INVOKER, then the procedure runs with the privileges that the invoker has.

dbspaces

Any user that is granted the CREATE ON privilege on dbspaces must also have the appropriate CREATE privilege to create the objects they want to create.

Sequences

The USAGE privilege is the only privilege that is granted on sequence generators. This privilege allows users to evaluate the current or next value in a sequence. If the sequence is part of a DEFAULT clause on a table, any user that inserts a row into the table must have privilege on the sequence.

Related Information

Inheritance of Roles and Privileges [page 1617]
Microsoft Windows Integrated Login [page 129]
Kerberos User Authentication [page 159]
Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
Viewing the Roles and Privileges for a User or Role (SQL) [page 1613]
Granting an Object-level Privilege (SQL Central) [page 1601]

GRANT ROLE Statement
sp_objectpermission System Procedure

1.9.1.2.2 System Privileges

There are many system privileges that can be granted.

You can view the roles and privileges a user has from SQL Central by clicking them and viewing the details that are displayed. You can also retrieve the details by using the sp_displayroles system procedure.

> **i** Note
>
> By default, all system privileges are inherited by a user who has the SYS_RUN_REPLICATION_ROLE system role. This configuration is required because the SYS_AUTH_DBA_ROLE compatibility role is granted to SYS_RUN_REPLICATION_ROLE. You can revoke the SYS_AUTH_DBA_ROLE compatibility role from SYS_RUN_REPLICATION_ROLE to prevent this, and then grant only the privileges your replication requires to the SYS_RUN_REPLICATION_ROLE system role. However, a user granted the SYS_RUN_REPLICATION_ROLE system role can only exercise the privileges of this role on remote connections.

List of system privileges

System privilege according to object or functionality	Description	Users/roles can inherit from:
Databases		
ALTER DATABASE	Allows a user to: • Upgrade a database. • Perform cost model calibration. • Load database statistics. • Alter transaction logs (also requires the SERVER OPERATOR system privilege).	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
BACKUP DATABASE	Allows a user to back up a database.	• DIAGNOSTICS • SYS_AUTH_BACKUP_ROLE • SYS_AUTH_DBA_ROLE • SYS_AUTH_PROFILE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_SAMONITOR_ADMIN_ROLE
CHECKPOINT	Allows a user to force the database server to execute a checkpoint.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_SAMONITOR_ADMIN_ROLE
DROP CONNECTION	Allows a user to drop any connections to the database.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SSO_ROLE • SYS_RUN_REPLICATION_ROLE
MANAGE PROFILING	Allows a user to manage database server tracing.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_PROFILE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_RUN_PROFILER_ROLE
MONITOR	Allows a user to monitor a database, including accessing privileged statistics, connected users, and locks.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_SAMONITOR_ADMIN_ROLE • SYS_RUN_PROFILER_ROLE
Database options		

System privilege according to object or functionality	Description	Users/roles can inherit from:
SET ANY PUBLIC OPTION	Allows a user to set PUBLIC database options that do not require the SET ANY SECURITY OPTION or the SET ANY SYSTEM OPTION system privileges.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_REPLICATION_ADMIN_ROLE
SET ANY SECURITY OPTION	Allows a user to set any PUBLIC security database options.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SSO_ROLE • SYS_RUN_REPLICATION_ROLE
SET ANY SYSTEM OPTION	Allows a user to set PUBLIC system database options.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_REPLICATION_ADMIN_ROLE • SYS_RUN_PROFILER_ROLE
SET ANY USER DEFINED OPTION	Allows a user to set user-defined database options.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_REPLICATION_ADMIN_ROLE
Data types		
ALTER DATATYPE	Allows a user to alter data types.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
CREATE DATATYPE	Allows a user to create data types.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_RESOURCE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
DROP DATATYPE	Allows a user to drop data types.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
Dbspaces		
MANAGE ANY DBSPACE	Allows a user to: • Create, alter, drop, and comment on dbspaces. • Grant and revoke CREATE privileges on dbspaces. • Move data to any dbspace. • Run the database delete file function.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_SAMONITOR_ADMIN_ROLE
Debugging		
DEBUG ANY PROCEDURE	Allows a user to debug procedures, functions, triggers, and events.	• SA_DEBUG • SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
Events		
MANAGE ANY EVENT	Allows a user to create, alter, drop, trigger, and comment on events.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_SAMONITOR_ADMIN_ROLE • SYS_RUN_PROFILER_ROLE
External environments		
CREATE EXTERNAL REFERENCE	Allows a user to create external references in the database. You must have the system privileges required to create specific database objects before you can create external references. For example, creating a self-owned text configuration object that uses an external term breaker and/or prefilter requires both the CREATE TEXT CONFIGURATION and CREATE EXTERNAL REFERENCE system privileges.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_SAMONITOR_ADMIN_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
MANAGE ANY EXTERNAL ENVIRONMENT	Allows a user to alter, comment on, start, and stop external environments.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
MANAGE ANY EXTERNAL OBJECT	Allows a user to install, comment on, and remove external environment objects.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
Files, directories, and disk drives		
ACCESS DISK INFORMATION	Allows a user to access information regarding the total disk size and the remaining disk space by using the sp_disk_info system procedure.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
READ CLIENT FILE	Allows a user to read files on the client computer.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_READCLIENTFILE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
READ FILE	Allows a user to read files on the database server computer.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_READFILE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_RUN_PROFILER_ROLE
WRITE CLIENT FILE	Allows a user to write files to the client computer.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_AUTH_WRITECLIENTFILE_ROLE • SYS_RUN_REPLICATION_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
WRITE FILE	Allows a user to write files on the database server computer.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_AUTH_WRITE-FILE_ROLE • SYS_RUN_REPLICA-TION_ROLE • SYS_RUN_PRO-FILER_ROLE
Indexes		
ALTER ANY INDEX	Allows a user to alter and comment on indexes and text indexes on tables and views owned by any user.	• DIAGNOSTICS • SYS_AUTH_DBA_ROLE • SYS_AUTH_PRO-FILE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICA-TION_ROLE
CREATE ANY INDEX	Allows a user to create and comment on indexes and text indexes on tables and views owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICA-TION_ROLE
DROP ANY INDEX	Allows a user to drop indexes and text indexes on tables and views owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICA-TION_ROLE
LDAP		
MANAGE ANY LDAP SERVER	Allows a user to create, alter, drop, validate, and comment on LDAP servers.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SSO_ROLE • SYS_RUN_REPLICA-TION_ROLE
OData		
MANAGE ODATA	Allows a user to create, alter, and drop OData producer objects.	• SYS_RUN_REPLICA-TION_ROLE • SYS_AUTH_SA_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
VERIFY ODATA	Allows a user to be referenced in the AUTHENTICATION USER or ADMIN USER clause of the CREATE ODATA PRODUCER and ALTER ODATA PRODUCER statements. If a user without this privilege is referenced, then the OData producer returns an error.	• SYS_RUN_REPLICATION_ROLE • SYS_AUTH_SSO_ROLE
Materialized views		
ALTER ANY MATERIALIZED VIEW	Allows a user to alter and comment on materialized views owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
CREATE ANY MATERIALIZED VIEW	Allows a user to create and comment on materialized views owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
CREATE MATERIALIZED VIEW	Allows a user to create and comment on self-owned materialized views.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_RESOURCE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
DROP ANY MATERIALIZED VIEW	Allows a user to drop materialized views owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
Messages		
CREATE MESSAGE	Allows a user to create messages.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_RESOURCE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
DROP MESSAGE	Allows a user to drop messages.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
Miscellaneous		

System privilege according to object or functionality	Description	Users/roles can inherit from:
ALTER ANY OBJECT	Allows a user to alter and comment on the following types of objects owned by any user: • Data types • Events • Functions • Indexes • Materialized views • Messages • Procedures • Sequence generators • Spatial reference systems • Spatial units of measure • Statistics • Tables • Text configuration objects • Text indexes • Triggers • Views	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
ALTER ANY OBJECT OWNER	Allows a user to alter the owner of any type of object.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SSO_ROLE • SYS_RUN_REPLICATION_ROLE
COMMENT ANY OBJECT	Allows a user to comment on any type of object that can be created using the CREATE ANY OBJECT system privilege.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
CREATE ANY OBJECT	Allows a user to create and comment on the following types of objects owned by any user: • Data types • Events • Functions • Indexes • Materialized views • Messages • Procedures • Sequence generators • Spatial reference systems • Spatial units of measure • Statistics • Tables • Text configuration objects • Text indexes • Triggers • Views	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
DROP ANY OBJECT	Allows a user to drop the following types of objects owned by any user: • Data types • Events • Functions • Indexes • Materialized views • Messages • Procedures • Sequence generators • Spatial reference systems • Spatial units of measure • Statistics • Tables • Text configuration objects • Text indexes • Triggers • Views	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
MANAGE ANY OBJECT PRIVILEGE	Allows a user to: • Grant and revoke SELECT, INSERT, DELETE, UPDATE, ALTER, REFERENCES, LOAD, and TRUNCATE privileges on tables owned by any user. • Grant and revoke SELECT, INSERT, DELETE, and UPDATE privileges on views owned by any user. • Grant and revoke EXECUTE privileges on procedures and functions owned by any user. • Grant and revoke USAGE privileges on sequence generators owned by any user. • Grant and revoke CREATE privileges on dbspaces.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SSO_ROLE • SYS_REPLICATION_ADMIN_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_SAMONITOR_ADMIN_ROLE
MANAGE CACHED PLANS	Allows a user to access statistics related to cached plans on any connection.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
MANAGE CERTIFICATES	Allows a user to create, alter, drop, and comment on certificates.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SSO_ROLE • SYS_RUN_REPLICATION_ROLE
MANAGE TIME ZONE	Allows a user to create, alter, and drop simulated time zones.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
REORGANIZE ANY OBJECT	Allows a user to reorganize tables and materialized views.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
VALIDATE ANY OBJECT	Allows a user to validate tables, materialized views, indexes, and text indexes.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_AUTH_VALIDATE_ROLE • SYS_RUN_REPLICATION_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
Mirror servers		
MANAGE ANY MIRROR SERVER	Allows a user to: • Create, alter, drop, and comment on mirror servers. • Change mirror server parameters. • Set mirror server options.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
Mutexes and semaphores		• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE
CREATE ANY MUTEX SEMAPHORE	Allows a user to create a mutex or semaphore owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE
UPDATE ANY MUTEX SEMAPHORE	Allows a user to update a mutex or semaphore owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE
DROP ANY MUTEX SEMAPHORE	Allows a user to drop a mutex or semaphore owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE
Procedures		
ALTER ANY PROCEDURE	Allows a user to alter and comment on procedures and functions owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_SAMONITOR_ADMIN_ROLE • SYS_RUN_PROFILER_ROLE
CREATE ANY PROCEDURE	Allows a user to create and comment on procedures and functions owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_REPLICATION_ADMIN_ROLE • SYS_SAMONITOR_ADMIN_ROLE • SYS_RUN_PROFILER_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
CREATE PROCEDURE	Allows a user to create self-owned procedures and functions.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_RESOURCE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
DROP ANY PROCEDURE	Allows a user to drop procedures and functions owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_REPLICATION_ADMIN_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_RUN_PROFILER_ROLE
EXECUTE ANY PROCEDURE	Allows a user to execute procedures and functions owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
Auditing		
MANAGE AUDITING	Allows a user to run the following auditing-related system procedures: • sa_audit_string • sp_trace_events (with the include_audit_events bit set) • sp_trace_event_fields (with the include_audit_events bit set) • sp_trace_event_session_events (with the include_audit_events bit set) • sp_trace_event_session_targets (with the include_audit_events bit set) • sp_trace_event_session_target_options (with the include_audit_events bit set)	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SSO_ROLE • SYS_RUN_REPLICATION_ROLE
Email		

System privilege according to object or functionality	Description	Users/roles can inherit from:
SEND EMAIL	Allows a user to run the following mail-related system procedures: • xp_get_mail_error_code • xp_get_mail_error_text • xp_startmail • xp_stopmail • xp_startsmtp • xp_stopsmtp • xp_sendmail	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
Replication		
MANAGE REPLICATION	Allows a user to: • Create, alter, drop, and comment on publications. • Create, alter, and drop MobiLink users. • Create, alter, drop, start, stop, and synchronize SQL Remote and synchronization subscriptions. • Create, alter, drop, and comment on synchronization profiles. • Create, alter, drop, and comment on SQL Remote message types. • Set remote options. • Start and stop schema synchronization.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_REPLICATION_ADMIN_ROLE • SYS_RUN_REPLICATION_ROLE
Roles		
MANAGE ROLES	Allows a user to create new roles and act as a global administrator for new and existing roles. By default, MANAGE ROLES is granted administrative rights on each newly created role. Administration of a role can also be granted directly to users either during or after the creation of the role. When granted directly to a user, the user does not require the MANAGE ROLES system privilege to administer the role.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SSO_ROLE • SYS_REPLICATION_ADMIN_ROLE • SYS_RUN_REPLICATION_ROLE
UPGRADE ROLE	Allows a user to be a default administrator of any system privilege that is introduced when upgrading a SQL Anywhere database from version 16.0.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
Sequences		

System privilege according to object or functionality	Description	Users/roles can inherit from:
ALTER ANY SEQUENCE	Allows a user to alter sequence generators owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
CREATE ANY SEQUENCE	Allows a user to create sequence generators, regardless of owner.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_RESOURCE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
DROP ANY SEQUENCE	Allows a user to drop sequence generators owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
USE ANY SEQUENCE	Allows a user to use sequence generators owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
Server operators		
MANAGE ANY PROPERTY HISTORY	Allows a user to turn on and configure the tracking of database server property values.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
MANAGE LISTENERS	Allows a user to start and stop network listeners.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
SERVER OPERATOR	Allows a user to: • Create, restore, drop, start, and stop databases, change ownership of a database, and restore the catalog (only). • Create, alter, and drop remote servers and directory access servers. • Manage a server cache. • Start and stop database servers. • Create encrypted and decrypted databases and files. • Change a database transaction log.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_REPLICATION_ADMIN_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_SAMONITOR_ADMIN_ROLE • SYS_RUN_PROFILER_ROLE
Spatial objects		
MANAGE ANY SPATIAL OBJECT	Allows a user to create, alter, drop, and comment on spatial reference systems and spatial unit of measures.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_SPATIAL_ADMIN_ROLE
Statistics		
MANAGE ANY STATISTICS	Allows a user to create, alter, drop, and update database statistics for any table.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
Tables		
ALTER ANY TABLE	Allows a user to: • Alter and comment on tables (including proxy tables) owned by any user. • Comment on columns in tables (including proxy tables) owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
CREATE ANY TABLE	Allows a user to: • Create and comment on tables (including proxy tables) owned by any user. • Comment on columns in tables (including proxy tables) owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_REPLICATION_ADMIN_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_SAMONITOR_ADMIN_ROLE • SYS_RUN_PROFILER_ROLE
CREATE PROXY TABLE	Allows a user to create self-owned proxy tables.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_RESOURCE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
CREATE TABLE	Allows a user to create self-owned tables.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_RESOURCE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
DELETE ANY TABLE	Allows a user to delete rows in tables and views owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_RUN_PROFILER_ROLE
DROP ANY TABLE	Allows a user to drop tables (including proxy tables) owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_REPLICATION_ADMIN_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_RUN_PROFILER_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
INSERT ANY TABLE	Allows a user to insert rows into tables and views owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_RUN_PROFILER_ROLE
LOAD ANY TABLE	Allows a user to load data into tables owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
SELECT ANY TABLE	Allows a user to query tables and views owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_REPLICATION_ADMIN_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_RUN_PROFILER_ROLE
TRUNCATE ANY TABLE	Allows a user to truncate data for tables and materialized views owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
UPDATE ANY TABLE	Allows a user to update rows in tables and views owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_RUN_PROFILER_ROLE
Text configuration objects		
ALTER ANY TEXT CONFIGURATION	Allows a user to alter and comment on text configuration objects owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
CREATE ANY TEXT CONFIGURATION	Allows a user to create and comment on text configuration objects owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
CREATE TEXT CONFIGURATION	Allows a user to create self-owned text configuration objects.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_RESOURCE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
DROP ANY TEXT CONFIGURATION	Allows a user to drop text configuration objects owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
Tracing		
MANAGE ANY TRACE SESSION	Allows a user to perform all event tracing-related operations, with the exception of triggering a user trace event.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_RUN_PROFILER_ROLE
NOTIFY TRACE EVENT	Allows a user to trigger a user trace event.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_RUN_PROFILER_ROLE
Triggers		
ALTER ANY TRIGGER	Allows a user to alter and comment on triggers on tables and views.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_RESOURCE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
CREATE ANY TRIGGER	Allows a user to create and comment on triggers on tables and views.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_RESOURCE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
Users and login management		
ACCESS USER PASSWORD	Allows a user to access views that contain password hashes, and perform operations that involve accessing passwords, such as unloading, extracting, or comparing databases.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SSO_ROLE • SYS_RUN_REPLICATION_ROLE
CHANGE PASSWORD	Allows a user to manage user passwords for any user. This system privilege can apply to all users, or it can be limited to a set of specified users, or users who are granted one or more specified roles. This system privilege is not required to change a user's own password.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SSO_ROLE • SYS_RUN_REPLICATION_ROLE
MANAGE ANY LOGIN POLICY	Allows a user to create, alter, drop, and comment on login policies.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SSO_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_SAMONITOR_ADMIN_ROLE
MANAGE ANY USER	Allows a user to: • Create, alter, drop, and comment on database users (including assigning an initial password). • Force a password change on next login for any user. • Assign and reset the login policy for any user. • Create, drop, and comment on integrated logins and Kerberos logins. • Create and drop external logins.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SSO_ROLE • SYS_REPLICATION_ADMIN_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_SAMONITOR_ADMIN_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
OFFLINE RESET PASSWORD	Allows the user to reset another user's password when using the database server -orp option. The user to perform the reset should not be the DBA user.	• SYS_OFFLINE_RESET_PASSWORD_ROLE
SET USER	Allows a user to temporarily assume the roles and privileges of another user. This system privilege can apply to all users, or can be limited to a set of specified users, or users who are granted one or more specified roles.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_AUTH_SSO_ROLE • SYS_RUN_REPLICATION_ROLE
Variables		
CREATE DATABASE VARIABLE	Allows a user to create, select from, update, and drop self-owned database-scope variables.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
MANAGE ANY DATABASE VARIABLE	Allows a user to create and drop database-scope variables owned by self or by PUBLIC.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
SELECT PUBLIC DATABASE VARIABLE	Allows a user to select the value of a database-scope variable owned by PUBLIC.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
UPDATE PUBLIC DATABASE VARIABLE	Allows a user to update database-scope variables owned by PUBLIC.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
Views		
ALTER ANY VIEW	Allows a user to alter and comment on views owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE

System privilege according to object or functionality	Description	Users/roles can inherit from:
CREATE ANY VIEW	Allows a user to create and comment on views owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
CREATE VIEW	Allows a user to create self-owned views.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_RESOURCE_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
DROP ANY VIEW	Allows a user to drop views owned by any user.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE
Web services		
MANAGE ANY WEB SERVICE	Allows a user to create, alter, drop, and comment on web services.	• SYS_AUTH_DBA_ROLE • SYS_AUTH_SA_ROLE • SYS_RUN_REPLICATION_ROLE • SYS_REPLICATION_ADMIN_ROLE

Related Information

Security Considerations with Role-based Access Control and Synchronization
Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
Viewing the Roles and Privileges for a User or Role (SQL) [page 1613]
Alphabetical List of Database Options [page 663]

1.9.1.2.3 How User Privileges Are Assessed

Roles introduce complexities in the privileges of individual users.

Suppose user M_Haneef has SELECT and UPDATE privileges on a specific table, but is also a member of two roles. Suppose one of these roles has no access to the table at all, and one role has only SELECT privilege. What are the privileges in effect for this user?

If the user ID has been explicitly granted the appropriate privilege(s) to perform the action, then the action proceeds.

Otherwise, permission to perform the action depends on the privileges of each of the roles to which the member belongs. If any of these roles has the privilege to perform the action, the user ID has privilege by virtue of membership in that role, and the action proceeds.

1.9.1.2.4 Granting a System Privilege (SQL Central)

Grant a system privilege to authorize a user or role to perform an operation on the database.

Prerequisites

You must have the administrative rights on the system privilege you want to grant.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. Double-click *System Privileges* and select a system privilege.
3. In the right pane, click the *Grantees* tab.
4. Right-click anywhere in the pane, and then choose *New* > *Grantees*.
5. Select the user or role that you want to grant the system privilege to.
6. Click *OK*.

   A new row is added to the *System Privileges* table.
7. (Optional) Use the *Adm.* (administrative rights) column to change the administrative rights for the system privilege. A checkmark in the column means the user or role can administer (grant or revoke) the system privilege. An empty column indicates that the user or role does not have the administrative rights for the system privilege. Click in the column to toggle the option.

   (Optional) Use the *Exe.* (exercise rights) column to change the exercise rights for the user or role. A checkmark in the column means the user or role can exercise the system privilege. The exercise privilege is granted by default. Click in the column to toggle the option.
8. (Optional) If you granted the CHANGE PASSWORD or SET USER system privilege, then the *Options* column contains *All users*. The *Options* column only applies to these two system privileges.
   - If you granted the CHANGE PASSWORD system privilege, then by default the user or role can change the password for any user. To restrict the users whose passwords can be changed, right-click the row and click *Set Options*.
   - If you granted the SET USER system privilege, then by default the user or role can impersonate any user. To restrict the users who can be impersonated, right-click the row and click *Set Options*. The *Options* column only applies to the Change Password and Set User system privileges.

9. Click *File* > *Save*.

Results

The user or role is granted the system privilege(s). If you granted a system privilege to a role, then any user or role that is granted that role inherits the new system privilege.

Related Information

Inheritance of Roles and Privileges [page 1617]
Changes to the GRANT Statement Syntax [page 1664]
Granting an Object-level Privilege (SQL Central) [page 1601]
Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]

1.9.1.2.5 Granting a System Privilege (SQL)

Grant a system privilege to authorize a user or role to perform an operation on the database.

Prerequisites

You must have administrative rights on the system privilege you want to grant.

Procedure

1. Connect to the database.
2. Execute a statement similar to the following:

Option	Action
Grant a privilege, with no administrative rights.	`GRANT privilege-name` `TO userid;`
Grant a privilege, with administrative rights.	`GRANT privilege-name` `TO userid WITH ADMIN OPTION;`

Option	Action
Grant administrative rights only on a privilege.	`GRANT privilege-name` `TO userid WITH ADMIN ONLY OPTION;`

Results

The user or role is granted the system privilege. If you granted a system privilege to a role, then any user or role that is granted that role inherits the new system privilege.

Example

To grant the CREATE ANY OBJECT system privilege to the role RoleA without giving RoleA administrative rights, execute the following statement:

```
GRANT CREATE ANY OBJECT TO RoleA
```

To grant RoleA the CREATE ANY OBJECT system privilege along with the ability to grant or revoke the system privilege to and from other users and roles, execute the following statement:

```
GRANT CREATE ANY OBJECT TO RoleA WITH ADMIN OPTION
```

To give RoleA administrative rights to the BACKUP DATABASE system privilege, but not the ability to use the BACKUP DATABASE privilege, execute the following statement:

```
GRANT BACKUP DATABASE TO RoleA WITH ADMIN ONLY OPTION
```

To grant RoleA administrative rights (only) for the BACKUP DATABASE and CHECKPOINT system privileges, execute the following statement:

```
GRANT BACKUP DATABASE, CHECKPOINT TO RoleA WITH ADMIN ONLY OPTION;
```

Related Information

Inheritance of Roles and Privileges [page 1617]
Changes to the GRANT Statement Syntax [page 1664]
Granting an Object-level Privilege (SQL Central) [page 1601]
Viewing the Roles and Privileges for a User or Role (SQL) [page 1613]

1.9.1.2.6 Granting an Object-level Privilege (SQL Central)

Grant object-level privileges to allow access to a specific object, such as a table, view, procedure, sequence, or dbspace.

Prerequisites

To grant an object-level privilege, one of the following must be true:

- You are the owner of the database object you are granting privileges for.
- You have been granted the privilege and have administrative rights on it.
- You have the MANAGE ANY OBJECT PRIVILEGE system privilege.

Context

When you grant privileges to roles, they are inherited by the members of the role.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, double-click *Users* and select a user.
3. In the right pane, click the appropriate privileges tab. For example, to grant privileges on a table click the *Table Privileges* tab.

   The tab lists the privileges that the user has been explicitly granted.
4. Right-click anywhere in the pane and click *New* > *Privileges*.
5. Select the database objects(s) that you want to grant privileges on and then click *OK*.

   The database object is added as a row to the tab. By default, all privileges for the database object are granted.
6. (Optional) To modify the privileges granted to the user on a database object, click the relevant cell in the table that appears in the appropriate *Privileges* tab (see the legend for cell definitions).

   A checkmark grants the privilege to the user. A checkmark with a plus sign (+) grants the privilege to the user and grants the user the privilege to grant the same privilege to other users. A checkmark with a plus sign (+) is equivalent to executing the GRANT statement with the WITH GRANT OPTION clause.

   In the *Table Privileges* tab, when you select a table, the corresponding *Column Privileges* table appears below. Modifying a column privilege causes the *Column privileges are granted/granted with grant option* icon to appear in the *Table Privileges* table. Modifying table privileges affects whether specific column privileges can be granted.

→ Tip

- To undo your changes, select one or more rows on the *Privileges* tab and click *Edit* > *Undo*.
- To turn the *Legend* on or off, click *File* > *Show Legend*.

7. Click *File* > *Save*.

Results

The user is granted the specified object-level privilege(s).

Related Information

Object-level Privileges [page 1573]
Revoking an Object-level Privilege (SQL Central) [page 1606]
Granting an Object-level Privilege (SQL) [page 1602]
Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
GRANT Statement
GRANT statement (SQL Anywhere 12.0.1)

1.9.1.2.7 Granting an Object-level Privilege (SQL)

Grant object-level privileges to allow access to a specific object, such as a table, view, procedure, sequence, or dbspace.

Prerequisites

To grant an object-level privilege, one of the following must be true:

- You are the owner of the database object you are granting privileges for.
- You have been granted the privilege and have administrative rights on it.
- You have the MANAGE ANY OBJECT PRIVILEGE system privilege.

Context

When you grant privileges to roles, they are inherited by the members of the role.

Procedure

1. Connect to the database.
2. Execute a GRANT statement.

Results

The user is granted the specified object-level privilege.

Example

The following statement grants SELECT privilege on table myTable to user AnnW:

```
GRANT SELECT ON myTable TO AnnW;
```

The following statement grants SELECT privilege on table myTable to user AnnW, and gives her administrative rights for the privilege:

```
GRANT SELECT ON myTable TO AnnW WITH GRANT OPTION;
```

Related Information

Object-level Privileges [page 1573]
Granting an Object-level Privilege (SQL Central) [page 1601]
Revoking an Object-level Privilege (SQL) [page 1607]
Viewing the Roles and Privileges for a User or Role (SQL) [page 1613]
GRANT Statement

1.9.1.2.8 Revoking a System Privilege (SQL Central)

Revoke a system privilege from a user, user-extended role, or user-defined role.

Prerequisites

You must have administrative rights for the system privilege.

You cannot revoke privileges from the SYS role. You can revoke privileges from other system roles, provided that these privileges are not the default privileges for the system role.

Context

If a user has been granted exercise and administrative rights for a privilege, you can revoke the administration rights, or the exercise and administrative rights. You cannot revoke the exercise right (only), and leave the administrative right. To do this, revoke the system privilege entirely, and then regrant the system privilege with administrative rights only.

You cannot revoke system privileges from a compatibility role. However, you can migrate a compatibility role to a user-defined role, and then revoke privileges from the user-defined role.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *System privileges*, and then a system privilege.
3. Click the *Grantees* tab.
4. Choose one of the following:

Option	Action
Revoke the privilege including its administrative rights	Right-click the user or role and click *Delete*.
Revoke only the administrative rights	Click in the *Adm.* (administrative rights) column to remove the checkmark.
Revoke only the privilege	Click in the *Exe.* (exercise rights) column to remove the checkmark.

→ Tip

- To undo your changes, select one or more rows and click *Edit* > *Undo*.
- To turn the *Legend* on or off, click *File* > *Show Legend*.

5. Click *File* > *Save*.

Results

The system privilege and/or the administrative right is revoked from the user or role.

Related Information

Inheritance of Roles and Privileges [page 1617]
Migrating a Compatibility Role to a User-defined Role (SQL Central) [page 1535]
Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
REVOKE Statement
GRANT Statement

1.9.1.2.9 Revoking a System Privilege (SQL)

Revoke a system privilege from a user, user-extended role, or user-defined role.

Prerequisites

You must have administrative rights for the system privilege.

You cannot revoke privileges from the SYS role. You can revoke privileges from other system roles, provided that these privileges are not the default privileges for the system role.

Context

If a user has been granted exercise and administrative rights for a privilege, you can revoke the administration rights, or the exercise and administrative rights. You cannot revoke the exercise right (only), and leave the administrative right. To do this, revoke the system privilege entirely, and then regrant the system privilege with administrative rights only.

You cannot revoke system privileges from a compatibility role. However, you can migrate a compatibility role to a user-defined role, and then revoke privileges from the user-defined role.

Procedure

1. Connect to the database.
2. Execute a REVOKE statement similar to the following:

```
REVOKE privilege FROM grantee;
```

Results

The system privilege and/or the administrative right is revoked from the user or role.

Related Information

Inheritance of Roles and Privileges [page 1617]
Migrating a Compatibility Role to a User-defined Role (SQL Central) [page 1535]
Viewing the Roles and Privileges for a User or Role (SQL) [page 1613]
REVOKE Statement
GRANT Statement

1.9.1.2.10 Revoking an Object-level Privilege (SQL Central)

Revoke an object-level privilege that has been granted to a user or role to restrict access to a specific object, such as a table, view, procedure, sequence, or dbspace.

Prerequisites

To revoke an object-level privilege, one of the following must be true:

- You are the owner of the database object you are revoking privileges for.
- You have been granted the privilege and have administrative rights on it.
- You have the MANAGE ANY OBJECT PRIVILEGE system privilege.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, either click the database object that you want to revoke a user privilege from, or click *Users* or *Roles* and click the user or role that you want to revoke privileges from.

Option	Action
Revoke an object-level privilege per user or role	1. In the right pane click the privileges tab that corresponds to the database object that you want to revoke privileges on. For example, to revoke privileges on a table, click the *Table privileges* tab. 2. Right-click the privilege and click *Revoke* then either click *All Privileges* or the specific privilege you want to revoke.

Option	Action
Revoke an object-level privilege per database object	1. In the right pane click the *Privileges* tab. 2. Right-click the user or role and click *Revoke* then either click *All Privileges* or the specific privilege you want to revoke.

3. Click *File* > *Save*.

Results

The privilege is revoked from the specified user or role.

Related Information

Object-level Privileges [page 1573]
Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
REVOKE Statement

1.9.1.2.11 Revoking an Object-level Privilege (SQL)

Revoke an object-level privilege that has been granted to a user or role to restrict access to a specific object, such as a table, view, procedure, sequence, or dbspace.

Prerequisites

To revoke an object-level privilege, one of the following must be true:

- You are the owner of the database object you are revoking privileges for.
- You have been granted the privilege and have administrative rights on it.
- You have the MANAGE ANY OBJECT PRIVILEGE system privilege.

Procedure

1. Connect to the database.
2. Execute a REVOKE statement similar to the following:

```
REVOKE privilege-name ON object-name FROM userid;
```

Results

The privilege is revoked from the specified user or role.

Related Information

Object-level Privileges [page 1573]
Viewing the Roles and Privileges for a User or Role (SQL) [page 1613]
REVOKE Statement

1.9.1.3 Users

All new users are automatically granted the PUBLIC system role.

You can control the privileges granted automatically to new users by editing the privileges included in the PUBLIC role.

By default, the privileges assigned to new users include:

- The ability to connect to the database (assuming a password has been specified for the user).
- The ability to view the data stored in the system views.
- The ability to execute most system stored procedures.

Once you have created a new user, you can:

- Grant system roles and system privileges to the user.
- Set the user's object-level privileges.
- Assign a login policy to the user. By default, a user is assigned to the root login policy.
- Set the user as the publisher or as a remote user of the database for use in a SQL Remote system.

In this section:

Creating a User (SQL Central) [page 1609]
Create a user by using the *Create User Wizard* in SQL Central.

Creating a User (SQL) [page 1610]
Create a user by using the CREATE USER statement.

Best Practices When Creating Users [page 1611]
When creating a SQL Anywhere user, follow these best practices to ensure optimal security.

Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
View the roles and privileges have been granted to users and to roles.

Viewing the Roles and Privileges for a User or Role (SQL) [page 1613]
View the roles and privileges a user or has, including roles and privileges they are inheriting.

Deleting a User (SQL Central) [page 1614]

Delete a user, as well as all database objects (such as tables) that the user owns, and any external logins for the user.

Related Information

Super-users [page 1558]

1.9.1.3.1 Creating a User (SQL Central)

Create a user by using the *Create User Wizard* in SQL Central.

Prerequisites

You must have the MANAGE ANY USER system privilege.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, right-click *Users*, and then click *New* > *User*.
3. Follow the instructions in the wizard.

Results

A user is created.

Next Steps

Grant roles or privileges to the user.

Related Information

Password and User ID Restrictions and Considerations [page 1634]

Login Policies [page 640]
Converting a User to a User-extended Role (SQL Central) [page 1550]
Granting a Role (SQL Central) [page 1563]
Managing Connected Users (SQL Central) [page 15]
CREATE USER Statement

1.9.1.3.2 Creating a User (SQL)

Create a user by using the CREATE USER statement.

Prerequisites

You must have the MANAGE ANY USER system privilege.

Procedure

1. Connect to the database.
2. Execute a CREATE USER statement that contains an IDENTIFIED BY clause.

   By default, users are assigned the root login policy. If you need to assign a user a different login policy, you can specify it by using the LOGIN POLICY clause of the CREATE USER statement.

Results

A user is created and the specified login policy is assigned to the user. If no login policy was specified, the root login policy is applied.

Example

This example adds a new user to the database with the user ID of M_Haneef and a password of Welcome.

```
CREATE USER M_Haneef
IDENTIFIED BY Welcome;
```

Next Steps

Grant roles or privileges to the user.

Related Information

Password and User ID Restrictions and Considerations [page 1634]
Login Policies [page 640]
Converting a User to a User-extended Role (SQL Central) [page 1550]
CREATE USER Statement

1.9.1.3.3 Best Practices When Creating Users

When creating a SQL Anywhere user, follow these best practices to ensure optimal security.

- Use the min_password_length Option [page 772] to require long user passwords.
- Ensure users have the change_password_dual_control option enabled in their login policy. See Dual Control Passwords [page 1639].
- In the login policy, limit failed login attempts.
- In the login policy, lock users if no login attempts occur in a specified number of days.
- In the login policy, specify password lifetime.
- Use the verify_password_function [page 866] option to require complex passwords. This option can also be used to prevent password reuse.

Related Information

Plan and Implement a Role-based Security Hierarchy [page 1631]
Login Policies [page 640]

1.9.1.3.4 Viewing the Roles and Privileges for a User or Role (SQL Central)

View the roles and privileges have been granted to users and to roles.

Prerequisites

You must have the PUBLIC role.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. Choose one of the following options:

Option	Action
View the roles and privileges for a user	1. In the left pane, double-click *Users*. 2. Click the user you want to view. 3. To view the user's roles and system privileges, in the right pane, click *Roles* and *System Privileges*, respectively. 4. To view the user's object-level privileges, click any of the remaining privileges tabs (such as *Table privileges*, *View privileges*, and so on).
View the roles and privileges for a role	1. In the left pane, double-click *Roles*. 2. Click the role you want to view. 3. To view the roles and system privileges, in the right pane, click *Roles* and *System Privileges*, respectively. 4. To view the role's object-level privileges, click any of the remaining privileges tabs (such as *Table privileges*, *View privileges*, and so on).

3. Optionally, select *Show Inherited* to include inherited roles and privileges.

Results

The right pane displays the roles and privileges that have been granted to, or inherited by, the specified user or role.

Related Information

Viewing the Roles and Privileges for a User or Role (SQL) [page 1613]

sp_displayroles System Procedure
sp_objectpermission System Procedure

1.9.1.3.5 Viewing the Roles and Privileges for a User or Role (SQL)

View the roles and privileges a user or has, including roles and privileges they are inheriting.

Prerequisites

- **sp_displayroles system procedure**: No privileges are required to execute this procedure on yourself. However, to return the system privileges or roles for another user ID or a role, you must have the MANAGE ROLES system privilege.
- **sp_objectpermission system privilege**: No privileges are required to execute this procedure on yourself or on objects you own. However, to call this procedure on another user ID, or on an object owned by another user ID, you must have the MANAGE ANY OBJECT PRIVILEGE system privilege.

Context

You can also use this task to view the roles and privileges for a given role.

Procedure

1. Connect to the database
2. To view the roles and system privileges the user has, execute a statement that calls the sp_displayroles system procedure, similar to the following, where `userid` is the user ID of the user:

   ```
   SELECT * FROM sp_displayroles( 'userid', 'expand_down');
   ```
3. To view the object-level privileges a user has, execute a statement that calls the sp_objectpermission system procedure, similar to the following:

   ```
   SELECT * FROM sp_objectpermission( 'userid' );
   ```

Results

The role_name column in the results for sp_displayroles includes inherited roles and system privileges as well as those explicitly granted to the user. If the role or system privilege is inherited from another role, the name of

that role is indicated in the parent_role_name column. The grant_type column tells you if the user has administrative rights on the role or system privilege. The role_level column conveys a hierarchy for the inheritance, since inheritance can occur by being member of a role that is a member of another role, and so on. This can help you troubleshoot when you revoke a role or privilege from a user but find they are still able to use the role or privilege.

The results for sp_objectpermission includes inherited object-level privileges as well as privileges explicitly granted to the user. Use the grantee column to learn where the object-level privilege is inherited from. The grantor column tells you who performed the actual granting. The grantable column tells you whether the user has administrative rights on the privilege.

Related Information

Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
sp_displayroles System Procedure
sp_objectpermission System Procedure

1.9.1.3.6 Deleting a User (SQL Central)

Delete a user, as well as all database objects (such as tables) that the user owns, and any external logins for the user.

Prerequisites

You must have the MANAGE ANY USER system privilege.

The user being removed cannot be connected to the database when you perform this procedure.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. In the left pane, click *Users*.
3. In the right pane, right-click a user and then click *Delete*.
4. Click *Yes* to confirm your action.

Results

The user is deleted as well as all database objects (such as tables or procedures) that the user owned, and any external logins for the user. If the user was a user-extended role, then this role is revoke from all users and roles

that had been granted the user-extended role. In addition, if the user is specified in the USER clause of any services, then those services are also dropped.

Related Information

User-extended Roles [page 1548]
DROP USER Statement
REVOKE Statement

1.9.1.4 Groups

A **group** is a set of users that possess a set of roles and privileges common to all other users in the group.

Groups are an efficient way to maintain roles and privileges for a set of users.

Membership in a group that owns objects also means that users do not need to qualify the object names when performing operations such as executing a procedure owned by the group, or querying a table owned by the group. For example, if a table named MyData is owned by a group called PersonnelData that a user ID M_Haneef is a member of, then M_Haneef can reference the table as MyData in queries, instead of PersonnelData.MyData. If a user owns a table that has the same name as a table owned by a group, the database server uses the table owned by the user, not the one owned by the group. If a user belongs to more than one group that has a table with the same name, the user must qualify the table name.

A group is not a database object. Instead, groups are achieved using user-defined roles, by one of two approaches:

User-extended role approach to creating groups

User-extended roles are user-defined roles created by extending an existing user ID to become a role. In this approach to group creation, the user ID is created and then granted all roles and privileges required by the group. The user ID is converted to a user-extended role, and then granted to the users who need it. These steps can occur in different order, of course.

This approach is common when creating groups for replication purposes because the user that has been extended to a role also has login capabilities.

For users of previous versions of SQL Anywhere, this approach is identical to the deprecated GRANT statement approach of granting GROUP to a user, and then granting MEMBERSHIP IN GROUP to other users.

To drop a group created using this approach, you convert the user-extended role back to a regular user (`DROP ROLE FROM USER` *userid*). Ownership of objects remains with the user that is being converted back to a regular user.

When the user-extended role approach is used, groups and their member information can be queried from the SYSROLEGRANTS consolidated view, or the SYSGROUPS compatibility view.

Standalone role approach to creating groups

Standalone roles are user-defined roles that are not associated with any user ID. In this approach to group creation, the role is created and then granted all roles and privileges required by a defined set of users. The role is then granted to the users who need it.

To drop a group created using the standalone approach, you drop the role. Optionally, you can drop the objects owned by the role at the same time you drop the role.

When the standalone role approach is used, groups and their member information can not be queried from the SYSGROUPS compatibility view. Instead, the information can be queried from the SYSROLEGRANTS consolidated view.

For both approaches, administration of the privileges for group members is performed at the user-defined role level, instead of at the level of each user ID.

Example

The following SQL statements show a very simplistic example of creating a group that has three members who need the SELECT ANY TABLE system privilege, and the SA_DEBUG system role. The first set of statements show the user-extended approach; the second set of statements show the standalone role approach.

It can be said both groups have three members, even though in the case of the user-extended approach, one member (member1) is acting both as role and member.

```
CREATE USER member1 IDENTIFIED BY sql;
CREATE USER member2 IDENTIFIED BY sql;
CREATE USER member3 IDENTIFIED BY sql;
CREATE ROLE FOR USER member1;
GRANT ROLE member1 TO member2, member3;
GRANT SELECT ANY TABLE TO member1;
GRANT ROLE SA_DEBUG TO member1:
```

```
CREATE ROLE standaloneRole;
CREATE USER member1 IDENTIFIED BY sql;
CREATE USER member2 IDENTIFIED BY sql;
CREATE USER member3 IDENTIFIED BY sql;
GRANT ROLE standaloneRole TO member1, member2, member3;
GRANT SELECT ANY TABLE TO standaloneRole;
GRANT ROLE SA_DEBUG TO standaloneRole;
```

Related Information

User-defined Roles [page 1543]
Groups Are Now Achieved Using Roles [page 1662]
Creating a User-defined Role (SQL) [page 1547]
Creating a User-defined Role (SQL Central) [page 1546]
Converting a User to a User-extended Role (SQL) [page 1551]
Converting a User to a User-extended Role (SQL Central) [page 1550]
Granting a Role (SQL Central) [page 1563]
Granting a Role (SQL) [page 1565]

GRANT ROLE Statement
SYSROLEGRANTS Consolidated View
SYSGROUP Compatibility View

1.9.1.5 Inheritance of Roles and Privileges

The privileges that a role or user has can be grouped into several categories.

Privileges explicitly granted to a user or group (role)

These are the privileges that are explicitly set for a user or role to control whether they can create, execute, delete, modify, or use specified database objects.

You cannot revoke a privilege that was not explicitly granted. For example, suppose user BobW is a member of a role called Sales. If a user grants DELETE privilege on the table T to the Sales role, then BobW can delete rows from table T. To prevent BobW from deleting from table T, you cannot revoke DELETE on the table T from BobW, since the DELETE ON T privilege was never explicitly granted to BobW.

Privileges acquired through ownership of an object

A user who creates a new object within the database is called the owner of that object, and automatically has permission to perform any operation on that object. For example, the owner of a table can modify the structure of that table, or can grant permissions to other database users to update the information within the table.

If a user is a user-extended role, grantees of that role do not inherit the privileges associated with ownership of the objects. The privileges must be explicitly granted to them, or inherited by them through a role.

Privileges inherited through roles

Each user ID can be granted one or more roles. Grantees inherit the privileges associated with that role.

For object-level privileges, ownership of database objects is associated with a single user ID. When the owner is a user-extended role or standalone role, ownership of the database object is not inherited by its grantees. For example, if a user-extended role or a standalone role owns objects, grantees of the user-extended role or standalone role cannot automatically query the objects; they must still be given SELECT privilege to query the objects.

Privileges inherited through groups

Groups are achieved through user-defined roles, so the inheritance of privileges through group membership is the same as through roles.

Privileges granted on disabled objects

You can grant privileges on disabled objects. Privileges for disabled objects are stored in the database and become effective when the object is enabled. The inheritance hierarchy can have many levels if roles have been granted to roles.

In this section:

Several scenarios are provided to show how privileges are inherited, especially with regard to different grant levels of administration rights.

Related Information

How User Privileges Are Assessed [page 1597]
Granting an Object-level Privilege (SQL Central) [page 1601]
Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
Viewing the Roles and Privileges for a User or Role (SQL) [page 1613]
GRANT Statement
GRANT ROLE Statement

1.9.1.5.1 Inheritance of Roles

By default, system privileges and roles are inherited by grantees of the roles.

For backwards compatibility with pre-16.0 databases, however, it was necessary to allow the ability to block the inheritance of those roles which were previously non-inheritable authorities. These roles include:

- SYS_AUTH_DBA_ROLE (previously, the DBA authority)
- SYS_AUTH_RESOURCE_ROLE (previously, the RESOURCE authority)
- SYS_AUTH_BACKUP_ROLE (previously, the BACKUP authority)
- SYS_AUTH_VALIDATE_ROLE (previously, the VALIDATE authority)
- SYS_RUN_REPLICATION_ROLE (previously, the REMOTE DBA authority)

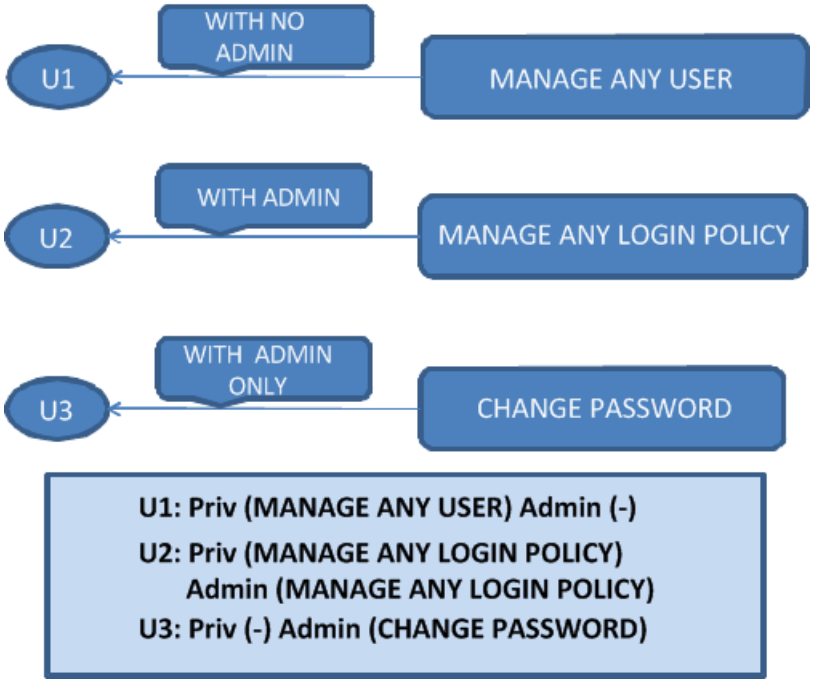
Related Information

Changes in Inheritance Behavior for Some Authorities That Became Roles [page 1675]
GRANT ROLE Statement

1.9.1.5.2 Inheritance Scenarios

Several scenarios are provided to show how privileges are inherited, especially with regard to different grant levels of administration rights.

Scenario 1



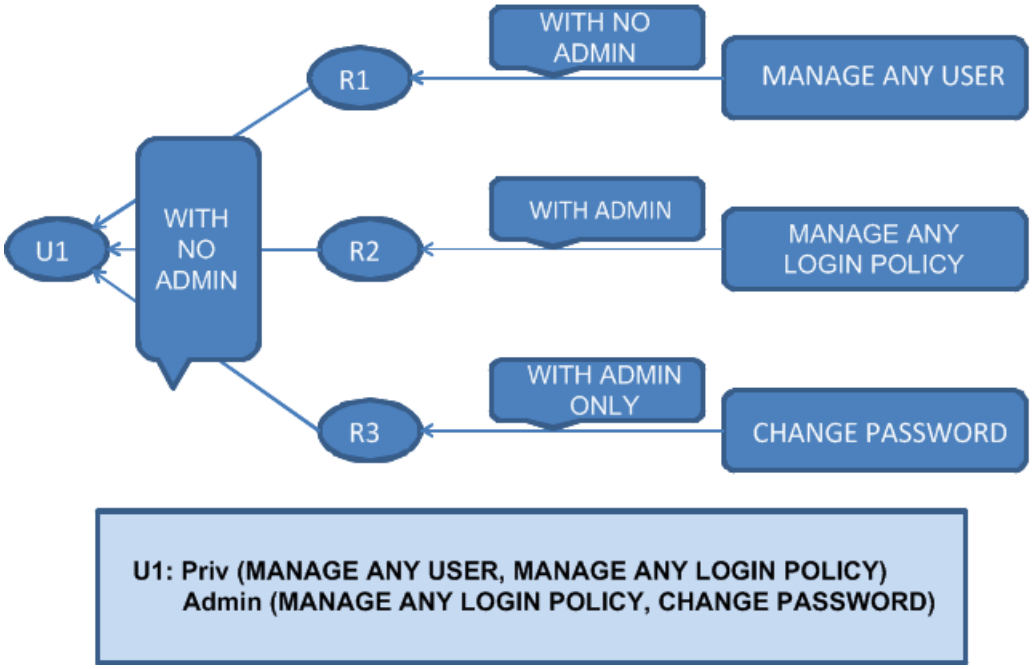
This diagram shows that:

- User 1 was granted the MANAGE ANY USER system privilege without administrative rights.
- User 2 was granted the MANAGE ANY LOGIN POLICY system privilege with administrative rights.
- User 3 was granted the CHANGE PASSWORD system privilege with administrative rights only.

From this, User 1, User 2, and User 3 have the following privileges:

	User 1	User 2	User 3
MANAGE ANY USER	Exercise	-	-
MANAGE ANY LOGIN POLICY	-	Exercise, administer	-
CHANGE PASSWORD	-	-	Administer

Scenario 2



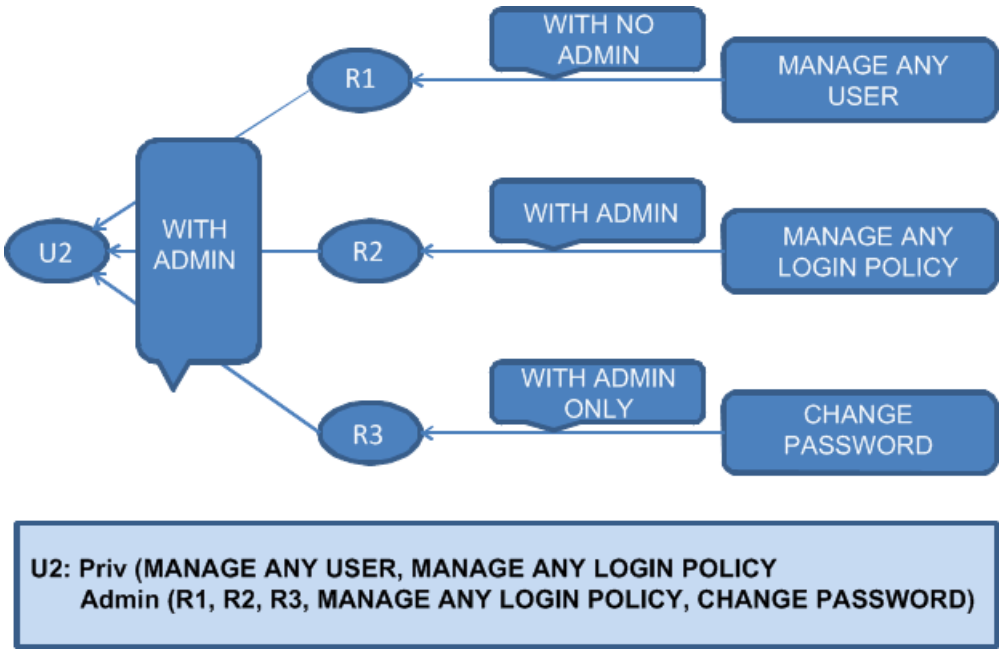
This diagram shows that:

- Role 1 was granted the MANAGE ANY USER system privilege without administrative rights.
- Role 2 was granted the MANAGE ANY LOGIN POLICY system privilege with administrative rights.
- Role 3 was granted the CHANGE PASSWORD system privilege with administrative rights only.
- User 1 was granted Role 1, Role 2, and Role 3 without administrative rights.

User 1 has the following privileges:

Role/Privilege	User 1
MANAGE ANY USER	Exercise
MANAGE ANY LOGIN POLICY	Exercise, administer
CHANGE PASSWORD	Administer
R1	-
R2	-
R3	-

Scenario 3



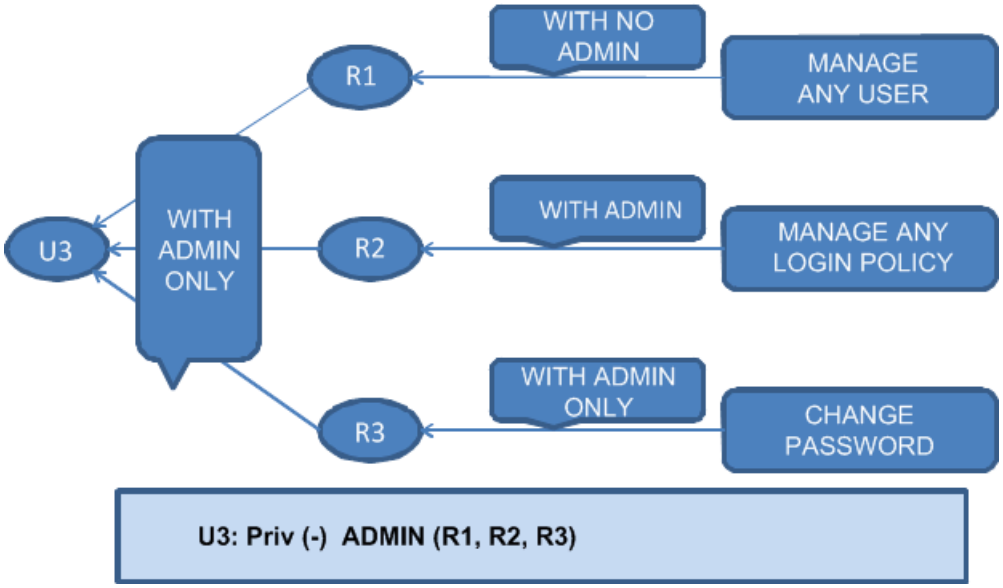
This diagram shows that:

- Role 1 was granted the MANAGE ANY USER system privilege without administrative rights.
- Role 2 was granted the MANAGE ANY LOGIN POLICY system privilege with administrative rights.
- Role 3 was granted the CHANGE PASSWORD system privilege with administrative rights only.
- User 2 was granted Role 1, Role 2, and Role 3 with administrative rights.

User 2 has the following privileges:

Role/Privilege	User 2
MANAGE ANY USER	Exercise
MANAGE ANY LOGIN POLICY	Exercise, administer
CHANGE PASSWORD	Administer
R1	Administer
R2	Administer
R3	Administer

Scenario 4



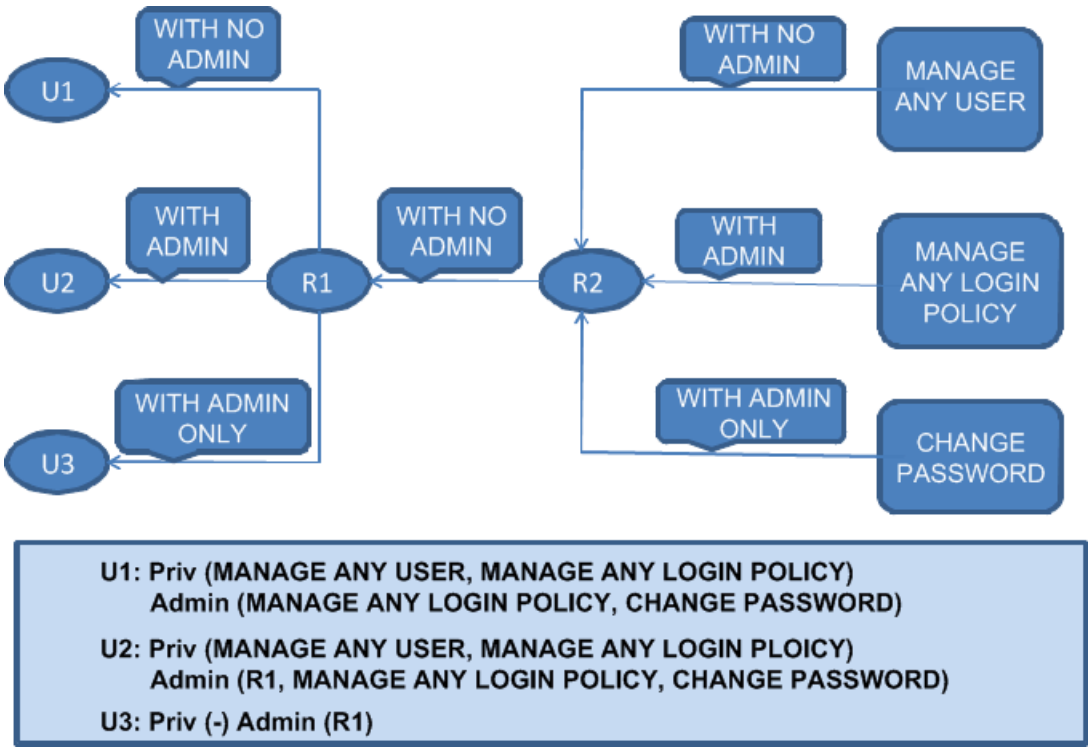
This diagram shows that:

- Role 1 was granted the MANAGE ANY USER system privilege without administrative rights.
- Role 2 was granted the MANAGE ANY LOGIN POLICY system privilege with administrative rights.
- Role 3 was granted the CHANGE PASSWORD system privilege with administrative rights only.
- User 3 was granted Role 1, Role 2, and Role 3 with administrative rights only.

User 3 has the following privileges:

Role/Privilege	User 3
MANAGE ANY USER	-
MANAGE ANY LOGIN POLICY	-
CHANGE PASSWORD	-
R1	Administer
R2	Administer
R3	Administer

Scenario 5



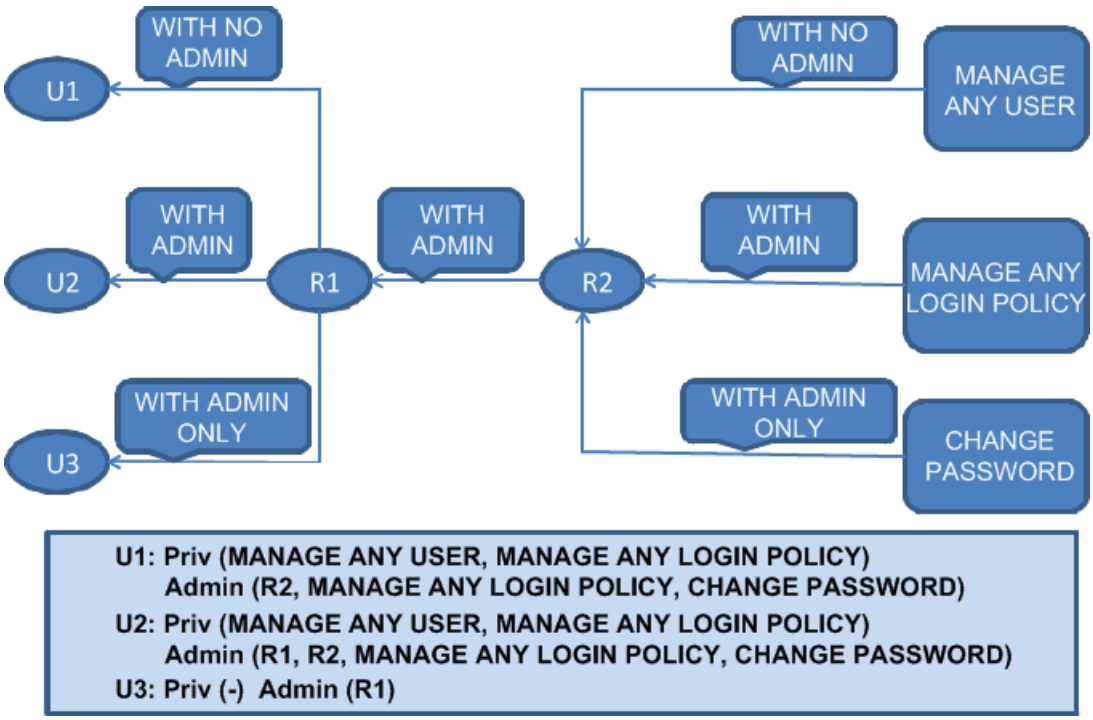
This diagram shows that:

- Role 2 was granted the MANAGE ANY USER system privilege without administrative rights.
- Role 2 was granted the MANAGE ANY LOGIN POLICY system privilege with administrative rights.
- Role 2 was granted the CHANGE PASSWORD system privilege with administrative rights only.
- Role 1 was granted Role 2 without administrative rights.
- User 1 was granted Role 1 without administrative rights.
- User 2 was granted Role 1 with administrative rights.
- User 3 was granted Role 1 with administrative rights only.

Users 1, 2 and 3 have the following privileges:

Role/Privilege	User 1	User 2	User 3
MANAGE ANY USER	Exercise	Exercise	-
MANAGE ANY LOGIN POLICY	Exercise, administer	Exercise, administer	-
CHANGE PASSWORD	Administer	Administer	-
R1	-	Administer	Administer
R2	-	-	-

Scenario 6



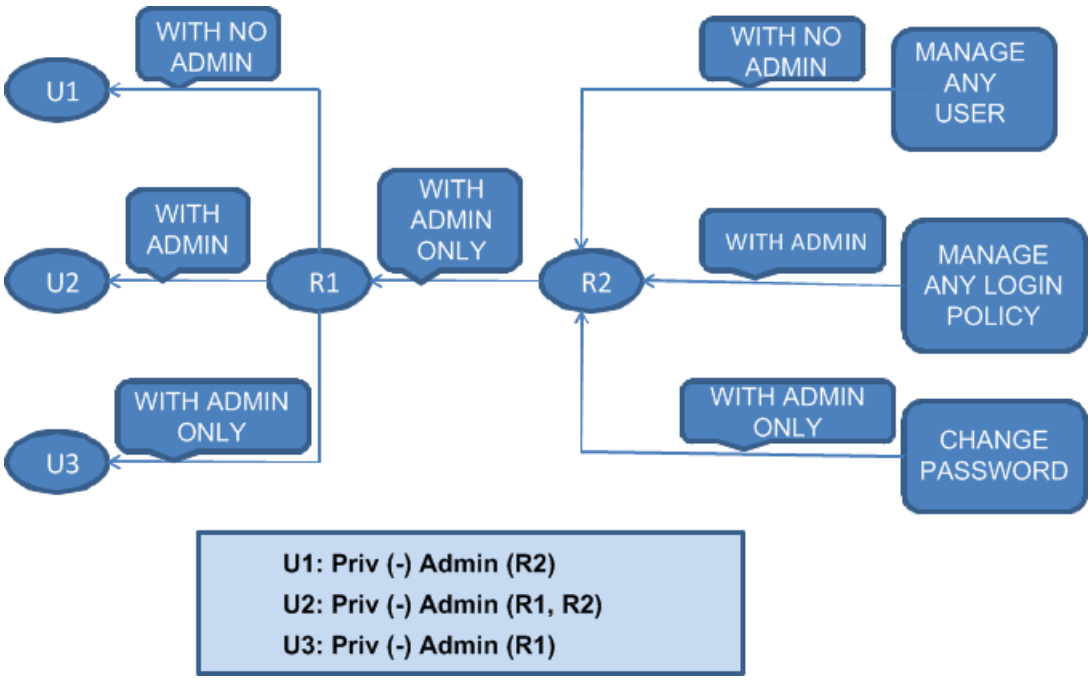
This diagram shows that:

- Role 2 was granted the MANAGE ANY USER system privilege without administrative rights.
- Role 2 was granted the MANAGE ANY LOGIN POLICY system privilege with administrative rights.
- Role 2 was granted the CHANGE PASSWORD system privilege with administrative rights only.
- Role 1 was granted Role 2 with administrative rights.
- User 1 was granted Role 1 without administrative rights.
- User 2 was granted Role 1 with administrative rights.
- User 3 was granted Role 1 with administrative rights only.

Users 1, 2 and 3 have the following privileges:

Role/Privilege	User 1	User 2	User 3
MANAGE ANY USER	Exercise	Exercise	-
MANAGE ANY LOGIN POLICY	Exercise, administer	Exercise, administer	-
CHANGE PASSWORD	Administer	Administer	-
R1	-	Administer	Administer
R2	Administer	Administer	-

Scenario 7



This diagram shows that:

- Role 2 was granted the MANAGE ANY USER system privilege without administrative rights.
- Role 2 was granted the MANAGE ANY LOGIN POLICY system privilege with administrative rights.
- Role 2 was granted the CHANGE PASSWORD system privilege with administrative rights only.
- Role 1 was granted Role 2 with administrative rights only.
- User 1 was granted Role 1 without administrative rights.
- User 2 was granted Role 1 with administrative rights.
- User 3 was granted Role 1 with administrative rights only.

Users 1, 2 and 3 have the following privileges:

Role/Privilege	User 1	User 2	User 3
MANAGE ANY USER	-	-	-
MANAGE ANY LOGIN POLICY	-	-	-
CHANGE PASSWORD	-	-	-
R1	-	Administer	Administer
R2	Administer	Administer	-

1.9.1.6 Impersonation

A user can temporarily assume the identity of another user in the database, also known as **impersonation**, to perform operations, provided they have a superset of the privileges of the person they are impersonating.

This restriction is referred to as the **at-least criteria**, and it extends to administrative rights and object-level privileges as well. If a user does not have *at least* the same privileges and administrative rights as the user they want to impersonate, they cannot impersonate them.

You might wonder why, if a user meets and even exceeds the at-least criteria, they don't just perform the operations themselves, instead of impersonating another user to do so. The reason is that if the impersonator has more privileges than required for the task, the additional privileges can affect the output of the task. Impersonating the user who normally performs the task negates this possibility. The goal is to recreate the privileges, and potentially the database options, that are in effect for the user who is being impersonated.

The ability to impersonate another user is controlled by the SET USER system privilege. When you grant the SET USER system privilege, you can configure who the user can impersonate to be one of the following:

- any user in the database
- users from a specified list of users
- users who are grantees of one or more of a specified list of roles

The at-least criteria is not evaluated at the time the SET USER system privilege is granted. Instead, it is evaluated when a user attempts to impersonate another user by executing a SETUSER statement.

While an impersonation session is in progress, any GRANT or REVOKE operations that would cause the at-least criteria to be violated are disallowed by the database server, and an error message is returned indicating the grant or revoke operation cannot proceed.

Example

Suppose a data entry clerk, JSmithClerk, is having difficulties performing an operation in the database. He is talking with PJonesIT in the IT department, and PJonesIT decides to impersonate JSmithClerk to observe and troubleshoot the problems that JSmithClerk is experiencing. In order for PJonesIT to impersonate JSmithClerk:

1. PJonesIT must have a superset of the privileges that JSmithClerk has.
2. PJonesIT must have been granted the SET USER system privilege in one of the following three ways:
   - impersonation rights directly over one or more users that includes JSmithClerk (`GRANT SET USER (JSmithClerk) TO PJonesIT;`)
   - impersonation rights over all users (`GRANT SET USER TO PJonesIT;`)
   - impersonation rights over users that have one or more of the roles that JSmithClerk has (`GRANT SET USER (WITH ROLES ...) TO PJonesIT;`)
3. PJonesIT must issue a SETUSER statement to initiate an impersonation session as JSmithClerk (`SETUSER JSmithClerk`). Any operations performed while the impersonation session is running are performed using the privileges of JSmithClerk. PJonesIT may decide that he wants the database options that are in effect for JSmithClerk to be applied to the impersonation session. In this case, PJonesIT includes the WITH OPTION clause in the SETUSER statement (`SETUSER WITH OPTION JSmithClerk;`).

Other Notes on Impersonation

- An impersonation session continues until the user performing the impersonation executes a SETUSER statement (without specifying a user), or until the connection ends, or until the user begins impersonating another user.
- In SQL Central, the *System Privileges* tab for a user provides details about who a user can impersonate on the SET USER row (if present). Hover the mouse over the *Options* cell to review the details. The details only include SET USER privileges granted directly to the user, not SET USER privileges they inherit.
- When a user impersonates another user and performs an operation, it is the user ID of the person being impersonated that is recorded in the transaction log for the operation. The SETUSER statement is also recorded in the transaction log, however, so an administrator can detect the start (and finish) of an impersonation session.
- If a user is not sure if they are still impersonating another user, they can execute `SELECT CURRENT USER`. The value returned is the user ID being used for the operation.
- In Interactive SQL, the title bar of the window indicates the user ID that is in effect when executing statements.

In this section:

Related Information

GRANT Statement
SETUSER Statement

1.9.1.6.1 In-depth Look at the Impersonation At-least Criteria

A user can successfully impersonate another user only if the at-least criteria are met.

Validation of the criteria occurs when a SETUSER statement is executed, not when the SET USER system privilege is granted. If a user fails to meet any of the criteria when the SETUSER statement is issued, the impersonation attempt fails, and an error is returned.

There are four criteria for successful impersonation:

1. The impersonator has been granted the right to impersonate the target user.
2. The impersonator has, at minimum, all the roles and system privileges granted to the target user.
3. The impersonator has been granted the roles and system privileges with similar or more administrative rights.
   For the purposes of meeting administrative rights criteria, the WITH ADMIN OPTION and WITH ADMIN ONLY OPTION clauses are considered to grant similar administrative rights. They are also considered to grant more administrative rights than the WITH NO ADMIN OPTION clause. For example, User1 is granted

Role1 with the WITH ADMIN OPTION clause, User2 is granted Role1 with the WITH ADMIN ONLY clause, and User3 is granted Role1 with the WITH NO ADMIN OPTION clause. User1 and User2 are considered to have Role1 with similar administrative rights. User1 and User2 are also considered to have Role1 with more administrative rights than User3.

4. If the target user has been granted a system privilege which supports additional parameters, also known as **extensions**, the clauses used to grant the system privilege to the impersonator are a superset of those used for the target user. For example, you can provide a list of user IDs when granting the SET USER system privilege.

   Grant the SET USER system privilege

   ```
   GRANT SET USER [ ( user-list | ANY [ WITH ROLES role-list ] ) ]
   TO grantee [,...]
   [ { WITH NO ADMIN | WITH ADMIN [ ONLY ] } OPTION ]
   ```

   Grant the CHANGE PASSWORD system privilege

   ```
   GRANT CHANGE PASSWORD [ ( user-list | ANY [ WITH ROLES role-list ] ) ]
   TO grantee [,...]
   [ { WITH NO ADMIN | WITH ADMIN [ ONLY ] } OPTION ]
   ```

   Currently, only the SET USER and CHANGE PASSWORD system privileges support extensions. The extensions specified when granting SET USER and CHANGE PASSWORD to the impersonator must be equal to, or a superset of, those specified when SET USER or CHANGE PASSWORD were granted to the target user. This is evaluated as follows:

   - The ANY extension is considered a superset to a `role-list` extension (list of roles) and a `user-list` extension (list of users). If the target user has the SET USER system privilege with the ANY extension (for example, `GRANT SET USER (ANY) TO user1`), the impersonator must also have the ANY extension.
   - If the target user has the SET USER system privilege with both the `role-list` and `user-list` extensions, the impersonator must also have the system privilege with the two extensions, and list for each extension must be equal to, or a superset to, the corresponding extension of the target user. For example, if the extension lists of both the impersonator and the target user contain User1, User2 and Role1, Role2, respectively, then the target list grants for each clause are considered equal. Alternately, if `user-list` for the impersonator contain User1, User2, Role1, Role2, respectively, while `user-list` for the target user contain User1, Role2 only, the impersonator's `user-list` is considered a superset to that of the target user.
   - If the target user has the SET USER system privilege with a single list extension, the extension list of the impersonator must be equal to, or a superset to, the list of the target user. For example, if the `user-list` of both the impersonator and the target user contain User1 and User2, they are considered equal. If `user-list` for the impersonator contains User1, User2, while `user-list` for the target user contains only User2, then `user-list` for the impersonator is considered to be a superset to `user-list` of the target user.

   So if the impersonator has the SET USER or CHANGE PASSWORD system privileges with extensions that limit who they can use them on, but the target user has the system privileges with no limitations, the impersonation will fail because the at-least criteria regarding extensions has not been met.

Example

Scenario 1 - Assuming that the second and third criterion is met, consider the following scenario:

- There are five users: User1, User2, User3, User4, and User5.
- There are two roles: Role1 and Role2.
- User1 was granted the SET USER system privilege with the ANY clause.
- User2 was granted the SET USER system privilege with the `user-list` clause for User1 and User4.
- User3 was granted the SET USER system privilege with the `user-list` clause for User1, User2, User4 and, User5, and the ANY WITH ROLES `role-list` clause for Role1 and Role2.
- User4 was granted the SET USER system privilege with the ANY clause and the `role-list` clause for Role1.
- User5 was granted the SET USER system privilege with the `user-list` clause for User4 and the ANY WITH ROLES `role-list` for Role1.
- User1 and User4 can successfully impersonate User2, User3, and User5 because each is granted the SET USER system privilege with the ANY clause. (Criteria 4).

User2, User3, and User5 cannot impersonate User1 or User4 because they do not have the ANY grant. (Criteria 4)

User2 cannot impersonate User3 or User5 because:

- User2 is not granted the right to impersonate these users. (Criteria 1)
- The SET USER system privilege is not granted to User2 with the `role-list` clause. (Criteria 4)

User3 can successfully impersonate User2 because:

- User3 is granted the right to impersonate User2 via the `user-list` clause. (Criteria 1)
- The `user-list` clause for User3 is a superset of User2. (Criteria 4) Though User3 has a grant with the `role-list` clause, it is not required to satisfy the requirements for impersonation of User2 because the latter does not have the same grant.

User3 can successfully impersonate User5 because:

- User3 is granted the right to impersonate User5 via the `user-list` clause. (Criteria 1)
- The `user-list` clause list for User3 is a superset of User5. (Criteria 4)
- The `role-list` clause lists for User3 and User5 are equivalent. (Criteria 4)

User5 cannot impersonate any other user because:

- User1 and User4 have an ANY grant (Criteria 4)
- User2 and User3 have a grant with a `user-list` clause that is not a sub-set of the grant to User5. (Criteria 4)

Scenario 2 - Assuming that the first and fourth criteria are met, consider the following:

- There are two users: User6 and User7.
- There are two roles: Role4 and Role5.
- User6 has been granted Role4 with the WITH ADMIN OPTION clause, Role5 with the WITH ADMIN ONLY OPTION clause, and the MANAGE ANY USER system privilege with the WITH ADMIN OPTION clause.
- User7 has been granted Role4 with the WITH ADMIN OPTION clause and Role5 with the WITH NO ADMIN OPTION clause.

User6 can successfully impersonate User7 because:

- Both User6 and User7 are granted Role4 and Role5. It does not matter that User6 is granted additional privileges (MANAGE ANY USER system privilege). (Criteria 2)
- User6 is granted Role4 with equivalent administrative rights as User7. User6 is granted Role5 with more administrative rights than User7. (Criteria 3)

User7 cannot impersonate User6 because:

- User7 is granted Role4 and Role5, but not the MANAGE ANY USER system privilege. (Criteria 2)
- User7 is granted Role5 with lower administrative rights than User6. (Criteria 3)

Scenario 3 - Consider the following:

- There are three users: User8, User9 and User10.
- There are two roles: Role5 and Role6.
- User8 has been granted Role5 with the WITH ADMIN OPTION clause, and the MANAGE ANY USER system privilege with the WITH ADMIN OPTION clause.
- User9 and User10 have been granted Role5 with the WITH NO ADMIN OPTION clause.
- User8 has been granted the SET USER system privilege to impersonate User9 and User10 with the `user-list` clause.
- User9 has been granted the SET USER system privilege to impersonate User10 with the `user-list` clause.

User8 can successfully impersonate User9 because:

- User8 is granted the right to impersonate User9 via the `user-list` clause. (Criteria 1)
- The `user-list` clause list for User8 is a superset of User9. (Criteria 4)
- Both User8 and User9 are granted Role5, with User8 granted more administrative rights to the role than User9. (Criteria 2 and 3)

User8 can successfully impersonate User10 because:

- User8 is granted the right to impersonate User10. (Criteria 1)
- Since User10 is not granted the SET USER system privilege, criteria 4 is not applicable.
- Both User8 and User10 are granted Role5, with the same administrative rights to the role. (Criteria 2 and 3)

User9 cannot impersonate User8 because:

- User9 is not granted the right to impersonate User8. (Criteria 1)
- Though both User8 and User9 are granted Role5, the grant for User9 is with fewer administrative rights to the role than for User8. (Criteria 3)

Validation of criteria occurs when the SETUSER statement is executed, not when the SET USER system privilege is granted. If a user fails to meet any of the criteria when the SETUSER statement is issued, a permission denied message appears, and the impersonation does not begin.

Related Information

GRANT Statement
SETUSER Statement

1.9.1.7 Plan and Implement a Role-based Security Hierarchy

A role-based security hierarchy needs to be planned and implemented.

Design the Security Hierarchy

1. Identify the various authorized tasks to be performed by users. Group closely related tasks. Groupings can be based on any organizational structure - departmental, functional, and so on. Assign a name to each grouping. These groupings are the **roles** you will create.
2. Identify the **system privileges** and **object-level privileges** required to perform each task that has been identified.
3. Identify the **users** that need to perform the various authorized tasks, including those who will require the roles you are going to create.
4. (Optional) Identify **administrators** for the roles you are going to create. Administrators can grant and revoke the role to other users.
5. (Optional) Identify administrators for the system privileges and object-level privileges that are not part of the roles you will be creating.

Build the Security Hierarchy

1. Create the required roles.
2. For each role, grant the required system privileges and object-level privileges.
3. Create the users.
4. Grant the applicable roles to the users, including granting administrative rights where applicable.
5. Grant any additional roles and privileges needed by individual users.

In this section:

Security: Control Privileges from the Command Line [page 1632]
Some database server options control the privileges required to perform certain global operations, including privileges to start and stop databases, load and unload data, and create and delete database files.

Related Information

Best Practices When Creating Users [page 1611]

1.9.1.7.1 Security: Control Privileges from the Command Line

Some database server options control the privileges required to perform certain global operations, including privileges to start and stop databases, load and unload data, and create and delete database files.

For example, the -gd database server option sets the privileges required to start or stop a database on a running database server.

Related Information

General Security Tips [page 1680]
Database Server Startup Options [page 390]

1.9.1.8 Ownership of Nested Objects

Views and procedures can access underlying objects that are owned by different users.

For example, if usera, userb, userc, and userd were four different users, userd.viewd could be based on userc.viewc, which could be based on userb.viewb, which could be based on usera.tablea. Similarly for procedures, userd.procd could call userc.procc, which could call userb.procb, which could insert into usera.tablea.

The following Discretionary Access Control (DAC) rules apply to nested views and tables:

- To create a view, the user must have the SELECT privilege on all the base objects (for example tables and views) in the view.
- To access a view, the view owner must have been granted the appropriate privilege on the underlying tables or views (with administration rights) and the user must have been granted the appropriate privilege on the view.
- Updating with a WHERE clause requires both the SELECT and UPDATE privileges.
- If a user owns the tables in a view definition, the user can access the tables through a view, even if the user is not the owner of the view and has not been granted access on the view.

The following DAC rules apply to nested procedures:

- A user does not require any privileges on the underlying objects (for example tables, views or procedures) to create a procedure.
- For a procedure to execute, the owner of the procedure needs the appropriate privileges on the objects that the procedure references.
- Even if a user owns all the tables referenced by a procedure, the user cannot execute the procedure to access the tables unless the user has been granted the EXECUTE privilege on the procedure.

Following are some examples that describe this behavior.

In this section:

1.9.1.8.1 Example 1: User1 Creates Table1, and User2 Creates View2 on Table1

User1 creates table1, and user2 creates view2 on table1.

- User1 can always access table1, since user1 is the owner.
- User1 can always access table1 through view2, since user1 is the owner of the underlying table. This is true even if user2 does not grant privilege on view2 to user1.
- User2 can access table1 directly or through view2 if user1 grants privilege on table1 to user2.
- User3 can access table1 if user1 grants privilege on table1 to user3.
- User3 can access table1 through view2 if user1 grants privilege on table1 to user2 with grant option *and* user2 grants privilege on view2 to user3.

1.9.1.8.2 Example 2: User2 Creates Procedure2 That Accesses Table1

User2 creates procedure2 that accesses table1.

- User1 can access table1 through procedure2 if user2 grants EXECUTE privilege on procedure2 to user1. This is different from the case of view2, where user1 did not need privilege on view2.

1.9.1.8.3 Example 3: User1 Creates Table1, User2 Creates Table2, and User3 Creates View3 Joining Table1 and Table2

User1 creates table1, user2 creates table2, and user3 creates view3 joining table1 and table2

- User3 can access table1 and table2 through view3 if user1 grants privilege on table1 to user3 *and* user2 grants privilege on table2 to user3.
- If user3 has privilege on table1 but not on table2, then user3 cannot use view3, even to access the subset of columns belonging to table1.
- User1 or user2 can use view3 if (a) user1 grants privilege with grant option on table1 to user3, (b) user2 grants privilege with grant option on table2 to user3, *and* (c) user3 grants privilege on view3 to that user.

1.9.1.9 Password and User ID Restrictions and Considerations

A user must have a password to connect to the database.

User IDs cannot:

- begin with white space, single quotes, or double quotes
- end with white space
- contain semicolons

Passwords are case-sensitive and they cannot:

- begin with white space, single quotes, or double quotes
- end with white space
- contain semicolons
- be longer than 255 bytes in length

By default, passwords must be 6 bytes in length. To change this requirement, set the min_password_length database option.

Additional rules for passwords can be set in a user's login policy.

When passwords are created or changed, they are converted to UTF-8, encrypted using a SHA256 hash, and then stored in the database. If the database is unloaded and reloaded into a database with a different character set, then existing passwords continue to work. If the database server cannot convert from the client's character set to UTF-8, then use 7-bit ASCII characters for the password as other characters may not work correctly.

Passwords are encrypted before they are sent from the client to the database server.

In this section:

Changing a Password (SQL Central) [page 1635]
Change the password for a user by editing the user properties.

Changing a Password (SQL) [page 1636]
Change the password for a user by using the ALTER USER statement with an IDENTIFIED BY clause.

Resetting the DBA Password (Command Line) [page 1637]
Reset the DBA user's password if the password is lost.

Dual Control Passwords [page 1639]
The dual control password feature requires two administrators to change a password.

Related Information

Login Policies [page 640]
min_password_length Option [page 772]

1.9.1.9.1 Changing a Password (SQL Central)

Change the password for a user by editing the user properties.

Prerequisites

Any user can change their own password.

To change the password for another user, you must have the CHANGE PASSWORD system privilege.

You cannot use SQL Central to change the password for another user who has dual control password changing enabled (that is, their change_password_dual_control login policy option is set to ON). If you are changing your own password and you have dual control password changing set to ON, you can still change your password normally; dual control password changing is not applied.

Context

A user has password and would like to change it.

Procedure

1. In SQL Central, use the *SQL Anywhere 17* plug-in to connect to the database.
2. Double-click *Users*, right-click a user and then click *Properties*.
3. Ensure that the *This user has a password* field is selected.
4. Complete the *Password* and *Confirm password* fields.
5. Click *Apply*, and then click *OK*.

Results

The user password is changed.

Next Steps

Inform the user of their new password.

Related Information

Login Policies [page 640]
Setting a Dual Control Password (SQL) [page 1639]
ALTER USER Statement

1.9.1.9.2 Changing a Password (SQL)

Change the password for a user by using the ALTER USER statement with an IDENTIFIED BY clause.

Prerequisites

Any user can change their own password. To change the password for another user, you must have the CHANGE PASSWORD system privilege.

If you are changing your own password and your login policy has dual control password changing set to ON, you can still change your password normally; dual control password changing is not required for you to change your own password.

Context

A user has a password and would like to change it.

Procedure

1. Connect to the database.
2. Execute an ALTER USER statement that contains an IDENTIFIED BY clause specifying the new password.

Results

The user password is changed.

Example

Execute the following statement to change the password for the DBA user to welcome_DBA:

```
ALTER USER DBA
IDENTIFIED BY welcome_DBA;
```

Next Steps

Inform the user of their new password.

Related Information

Setting a Dual Control Password (SQL) [page 1639]
ALTER USER Statement

1.9.1.9.3 Resetting the DBA Password (Command Line)

Reset the DBA user's password if the password is lost.

Prerequisites

- A role administrator has granted the SYS_OFFLINE_RESET_PASSWORD_ROLE system role to a database user.
- You must have access to the database server software.
- You must have read/write access to the database file.
- You must know the user ID and password of a database user that has been granted the SYS_OFFLINE_RESET_PASSWORD_ROLE system role.

Context

If there is another user that has been granted the CHANGE PASSWORD privilege, that user can change the DBA user's password provided the DBA user is not subject to the dual control password login policy. If the DBA user has the change_password_dual_control option enabled in their login policy, then a password change requires two different users that have been granted the CHANGE PASSWORD privilege to change the password. This is the preferred method for handling lost passwords. It does not require that the database be stopped.

Otherwise, the offline password reset task (described below) can be used to reset any user's password including those users subject to the dual control password login policy.

Procedure

1. Shut down the database.

   The database must be stopped to perform this task.
2. Start a database server with the -orp option and include a new password for the DBA user with the lost password. You must specify the user name and password of a database user that has been granted the SYS_OFFLINE_RESET_PASSWORD_ROLE system role.

   A message appears indicating whether the password reset was successful and then the database server shuts down.
3. Restart the database.
4. Inform the DBA user of their new password.

Results

The DBA user password is changed.

Example

The user `DBA` has the password sql123 and the user `reset_user` has the password sql456. As a role administrator, `DBA` grants the SYS_OFFLINE_RESET_PASSWORD_ROLE system role to `reset_user`:

```
GRANT ROLE SYS_OFFLINE_RESET_PASSWORD_ROLE TO reset_user;
```

The user `DBA` loses their password. The user `reset_user` resets the `DBA` user's password to be **`newpassword`**:

```
dbeng17 -orp "UID=DBA;NEWPWD=newpassword;AUTHUID=reset_user;AUTHPWD=sql456"
mydb.db
```

Related Information

System Roles [page 1528]
-orp Database Server Option [page 474]
System Roles [page 1528]
Super-users [page 1558]
-orp Database Server Option [page 474]

1.9.1.9.4 Dual Control Passwords

The dual control password feature requires two administrators to change a password.

This configuration prevents any single administrator from knowing the complete password of another user. One administrator sets the first part of a password, and another administrator sets the second part of the password. The user specifies the two password parts to connect to the database.

The target user must have the change_password_dual_control option enabled in their login policy.

In this section:

Related Information

ALTER USER Statement
Login Policy Options and Default Values [page 642]

1.9.1.9.4.1 Setting a Dual Control Password (SQL)

Set a dual control password or change the first or last part of a dual control password.

Prerequisites

You must have the CHANGE PASSWORD system privilege.

The target user must have the change_password_dual_control option enabled in their login policy.

Context

The dual control password feature prevents an administrator from knowing a user's complete password.

Another administrator must change the other part of the password. The user combines the two password parts and uses this combined password to connect to the database.

Password parts are case-sensitive and they cannot:

- begin with white space, single quotes, or double quotes
- end with white space
- contain semicolons

- be longer than 127 bytes in length

Procedure

1. Connect to the database.
2. An administrator executes the following statement to set the first part of the dual control password, and then communicates `password-part1` to the user:

   ```
   ALTER USER user-name IDENTIFIED FIRST BY password-part1;
   ```

3. Another administrator executes the following statement to set the second part of the dual control password, and then communicates `password-part2` to the user.

   ```
   ALTER USER user-name IDENTIFIED LAST BY password-part2;
   ```

Results

The password for the target user is set to use the combined first and last password parts (`password-part1 password-part2`).

Next Steps

The target user connects to the database using the combined parts as their password, and is prompted to create a new password.

Related Information

Login Policies [page 640]
ALTER USER Statement

1.9.1.10 Tutorial: Granting Roles and Privileges (SQL Central)

In the role-based security model, users must have the correct privileges and roles to perform specific database operations.

Prerequisites

None. You log into the database as the user ID DBA, who has all the system privileges and roles required for the tutorial.

Context

In this tutorial, you are a database administrator with full administrative capabilities. You want two colleagues to be responsible for debugging procedures and correcting any errors found during the debug process. You also want these same colleagues to be responsible for running backups on the company database (in this scenario, the database is the SQL Anywhere sample database). Finally, you want both colleagues to be able to update the Employees table in the database.

Create user IDs for both colleagues and grant them the privileges and roles necessary for them to do these tasks. In this tutorial, you use SQL Central to perform the tasks in the tutorial.

> **i** Note
>
> SQL Central caches information about the database such as what privileges and roles a user has. If you make changes to the database on another connection, SQL Central won't see the changes until you refresh the database in SQL Central. You can do this by selecting either the database or any object in the tree in the left panel and then pressing F5.

1. Lesson 1: Create Two New Users: User1 and User2 (SQL Central) [page 1642]
   Log in to the sample database using SQL Central and create two new users.
2. Lesson 2: Grant Privileges to User1 (SQL Central) [page 1643]
   Grant system privileges and object-level privileges to User1.
3. Lesson 3: Create a Role and Grant It to User1 (SQL Central) [page 1644]
   Create a role that contains the privileges necessary to debug a procedure and correct any errors found during the debugging process, and then grant that role to User1.
4. Lesson 4: Convert User1 to a User-extended Role and Grant That Role to User2 (SQL Central) [page 1646]
   Convert User1 to a user-extended role and grant that role to User2.
5. Lesson 5: View the Roles and Privileges for User2 (SQL Central) [page 1647]
   View the roles and privileges a user has, including those that the user inherited through membership in other roles.

Related Information

Upgrading from Authority-based Security to Role-based Security [page 1657]

1.9.1.10.1 Lesson 1: Create Two New Users: User1 and User2 (SQL Central)

Log in to the sample database using SQL Central and create two new users.

Procedure

1. Start SQL Central and connect to the database.
   a. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Central*.
   b. Click *Connections* > *Connect With SQL Anywhere 17*.
   c. In the *Password* field, type the password **`sql`**.
   d. In the *Action* dropdown list, select *Connect with an ODBC Data Source*.
   e. In the *ODBC Data Source name*, click *Browse*, select ***SQL Anywhere 17 Demo***, and then click *OK*.
   f. Click *Connect*. This connects you to the sample database as the user called DBA, with the password sql, and with full administrative rights on the database.
2. Create two new users.
   a. In the left pane, right-click *Users* and then click *New* > *User*.
   b. In the *What do you want to name the new user?* field, type **`User1`**, then click *Next*.
   c. Assign the password **`sql`** to User1, then click *Finish*.
   d. Repeat the above steps to create the second user, **`User2`**.

Results

You have created two new users, User1 and User2.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Granting Roles and Privileges (SQL Central) [page 1641]

Next task: Lesson 2: Grant Privileges to User1 (SQL Central) [page 1643]

Related Information

Creating a User (SQL Central) [page 1609]
Creating a User (SQL) [page 1610]
CREATE USER Statement

1.9.1.10.2 Lesson 2: Grant Privileges to User1 (SQL Central)

Grant system privileges and object-level privileges to User1.

Prerequisites

You must have completed the previous lessons in this tutorial.

Procedure

1. In the left pane in SQL Central, click to expand *Users*, and click **`User1`**.
2. Grant the BACKUP DATABASE and VALIDATE ANY OBJECT system privileges to User1.
   a. In the right pane, choose the *System Privileges* tab.
   b. Right-click in the tab, and click *New* > *Granted System Privileges*.
   c. Click the *BACKUP DATABASE* system privilege, then click *OK*.
   d. Repeat the above steps, this time clicking the *VALIDATE ANY OBJECT* system privilege, then click *OK*.
   e. Click *File* > *Save* to save your changes to the database.
3. Grant SELECT and UPDATE object-level privileges on the Employees table to User1, with administrative rights.
   a. In the right pane, choose the *Table Privileges* tab.
   b. Right-click in the tab, and click *New* > *Privileges*.
   c. Click the *Employees* table, then click *OK*. The Employees table is added to the list of tables for User1. Each check mark in the letter columns indicates an object-level privilege that User1 has on the table. For example, *S* for select, *I* for insert, and so on. However, in this lesson only User1 should have the ability to select from, and update, the Employees table.
   d. Click the cells beneath the columns to clear their contents until there is a check mark, with a plus sign (+), in only the *S* (for select) and *U* (for update) columns. The plus sign indicates administrative rights. All other cells in the row should be empty.
   e. Click *File* > *Save* to save your changes to the database.

Results

User1 now has the BACKUP DATABASE and VALIDATE ANY OBJECT system privileges, which are required to back up the SQL Anywhere sample database. User1 also has the SELECT and UPDATE object-level privileges on the Employees table, and can grant the privileges to other users.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Granting Roles and Privileges (SQL Central) [page 1641]

Previous task: Lesson 1: Create Two New Users: User1 and User2 (SQL Central) [page 1642]

Next task: Lesson 3: Create a Role and Grant It to User1 (SQL Central) [page 1644]

Related Information

System Privileges [page 1577]
Granting a System Privilege (SQL Central) [page 1598]
Granting a System Privilege (SQL) [page 1599]
GRANT Statement

1.9.1.10.3 Lesson 3: Create a Role and Grant It to User1 (SQL Central)

Create a role that contains the privileges necessary to debug a procedure and correct any errors found during the debugging process, and then grant that role to User1.

Prerequisites

You must have completed the previous lessons in this tutorial.

Procedure

1. Create the role.
   a. In the left pane in SQL Central, right-click *Roles* and click *New* > *Role*.
   b. In the *What do you want to name the new user-defined role?* field, type **DebugAndAlter**, and then click *Next*.
   c. On the *Choose the Administrators* screen, click *Finish* to accept the default, which is to allow users with the MANAGE ROLES system privilege (also known as global administrators) to administer the role.
2. Grant the required system privileges to the role.
   a. In the left pane, double-click *Roles* and select **DebugAndAlter**.
   b. In the right pane, click the *System Privileges* tab.
   c. Right-click in the tab and then click *New* > *Granted System Privileges*.
   d. Click the *ALTER ANY OBJECT* system privilege, then click *OK*.
   e. Repeat the steps to add the *DEBUG ANY PROCEDURE* system privilege.
   f. Click *File* > *Save* to save your changes to the database.
3. Grant the role to User1 with administrative rights on the role.
   a. In the left pane, double-click *Users* and select **User1**.
   b. In the right pane, click the *Roles* tab.
   c. Right-click in the tab, and then click *New* > *Granted Roles*.
   d. Click *DebugAndAlter* and then click *OK*. The ability to exercise the DebugAndAlter role is granted to User1, without administrative rights.
   e. In the right-pane, on the row for the DebugAndAlter role, click in the cell in the *Adm.* column to add a check mark. This grants administrative rights on the role to User1.
   f. Click *File* > *Save* to save your changes to the database.

Results

The role DebugAndAlter, which has the DEBUG ANY PROCEDURE and ALTER ANY OBJECT system privileges, has been created and granted to User1. User1 has also been given administrative rights on the role.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Granting Roles and Privileges (SQL Central) [page 1641]

Previous task: Lesson 2: Grant Privileges to User1 (SQL Central) [page 1643]

Next task: Lesson 4: Convert User1 to a User-extended Role and Grant That Role to User2 (SQL Central) [page 1646]

Related Information

Creating a User-defined Role (SQL Central) [page 1546]
Configuring Roles and Privileges For a Role (SQL Central) [page 1561]
Granting a Role (SQL Central) [page 1563]
GRANT ROLE Statement

1.9.1.10.4 Lesson 4: Convert User1 to a User-extended Role and Grant That Role to User2 (SQL Central)

Convert User1 to a user-extended role and grant that role to User2.

Prerequisites

You must have completed the previous lessons in this tutorial.

Context

Assume that you want User2 to have the same privileges and roles as User1. Rather than going through the same lengthy process of explicitly granting the roles and privileges as you did with User1, you can convert User1 to a user-extended role and grant that role to User2. User1 continues to be a user who can log in to the database, in addition to being a role.

Procedure

1. Convert User1 to a user-extended role.
   a. In the left pane, expand *Users*, right-click **User1**, and choose *Change to User-extended Role*.
2. Grant the user-extended role User1 to User2.
   a. In the left pane, in *Users*, select **User2**.
   b. In the right pane, choose the *Roles* tab.
   c. Right-click in the tab and choose *New* > *Granted Roles*.
   d. Click **User1**, then click *OK*.
   e. Click *File* > *Save* to save your changes to the database.

Results

User1 has been converted to a user-extended role and has been granted to User2.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Granting Roles and Privileges (SQL Central) [page 1641]

Previous task: Lesson 3: Create a Role and Grant It to User1 (SQL Central) [page 1644]

Next task: Lesson 5: View the Roles and Privileges for User2 (SQL Central) [page 1647]

Related Information

User-extended Roles [page 1548]
CREATE ROLE Statement
GRANT ROLE Statement

1.9.1.10.5 Lesson 5: View the Roles and Privileges for User2 (SQL Central)

View the roles and privileges a user has, including those that the user inherited through membership in other roles.

Prerequisites

You must have completed the previous lessons in this tutorial.

Procedure

1. In SQL Central, double-click *Users* in the left pane and click **`User2`**.
2. Ensure that the *Show Inherited* checkbox at the top of the screen is selected.

3. In the right pane, click the *Roles* tab to see the roles for User2:

   The screenshot shows that User2 has the following roles:

   dbo system role

   User2 has exercise rights, but not administrative rights, on the dbo system role. There is no grantor because User2 is inheriting this system role by being a member of the PUBLIC system role. Users are granted the PUBLIC system role by default when they are created.

   DebugAndAlter user-defined role

   User2 has exercise and administrative rights on the DebugAndAlter role. There is no grantor because User2 is inheriting this user-defined role from User1.

   PUBLIC system role

   User2 has exercise rights on the PUBLIC role. Users are granted the PUBLIC system role by default when they are created. The grantor is DBA because that is what you were logged in as when you created User2.

   User1 user-extended role

   User2 has exercise rights, but not administrative rights, on the User1 user-extended role. The grantor is DBA because that is what you were logged in as when you granted the role to User2.
4. Click the *System Privileges* tab to see the system privileges that User2 has. The *Grantor* column is empty for all of them because the system privileges were inherited, not granted explicitly to User2. The *Adm* column is empty because no administration rights were granted when the system privileges were granted.
5. Click the *Table Privileges* tab to view the object-level privileges on the Employees table that User2 inherited from User1. Although users can inherit object-level privileges, they cannot inherit administrative rights on object-level privileges. While User1 can exercise and administer SELECT and UPDATE privileges on the Employees table, User2 only inherits the abilities to SELECT from, and UPDATE, the table.

Results

You have verified the roles, privileges, and administrative rights for User2.

Task overview: Tutorial: Granting Roles and Privileges (SQL Central) [page 1641]

Previous task: Lesson 4: Convert User1 to a User-extended Role and Grant That Role to User2 (SQL Central) [page 1646]

Related Information

Inheritance of Roles and Privileges [page 1617]
Role Administrators [page 1555]
Granting a System Privilege (SQL Central) [page 1598]
Granting a Role (SQL Central) [page 1563]

1.9.1.11 Tutorial: Granting Roles and Privileges (SQL)

In the role-based security model, users must have the correct privileges and roles to perform specific database operations.

Prerequisites

None. You log into the database as user ID DBA, who has all the system privileges and roles required for the tutorial.

Context

In this tutorial, you are a database administrator with full administrative capabilities. You want two colleagues to be responsible for debugging procedures and correcting any errors found during the debug process. You also want these same colleagues to be responsible for running backups on the company database (in this scenario, the database is the SQL Anywhere sample database). Finally, you want both colleagues to be able to update the Employees table in the database.

You must create user IDs for both colleagues and grant them the privileges and roles necessary for them to do these tasks.

1. Lesson 1: Create Two Users: UserA and UserB (SQL) [page 1650]
   Log in as user ID DBA and create two users who will be responsible for backing up the sample database, debugging procedures, and correcting any mistakes found during the debug process.
2. Lesson 2: Grant Privileges to UserA (SQL) [page 1651]
   Grant system privileges and object-level privileges to UserA.
3. Lesson 3: Create a Role and Grant It to UserA (SQL) [page 1652]
   Create a role that contains the privileges necessary to debug a procedure and correct any errors found during the debugging process, then grant that role to UserA.
4. Lesson 4: Convert UserA to a User-extended Role and Grant That Role to UserB (SQL) [page 1653]
   Convert UserA to a user-extended role and then grant that role to UserB in SQL. UserA continues to be a user who can log in to the database, in addition to being a role.
5. Lesson 5: View the Roles and Privileges for UserB (SQL) [page 1655]
   View the roles, system privileges, and object-level privileges for a user, including those the user inherits through membership in other roles.

Related Information

Upgrading from Authority-based Security to Role-based Security [page 1657]

1.9.1.11.1 Lesson 1: Create Two Users: UserA and UserB (SQL)

Log in as user ID DBA and create two users who will be responsible for backing up the sample database, debugging procedures, and correcting any mistakes found during the debug process.

Procedure

1. In Interactive SQL, open the *Connect* window, and complete the following fields to connect to the sample database:
   a. In the *User ID* field, type **DBA**.
   b. In the *Password* field, type **sql**.
   c. In the *Action* dropdown list, select *Start and connect to a database on this computer*.
   d. In the *Server Name* field, type ***demo17***.
2. Create two new users.
   a. Execute the following statement to create UserA:

```
CREATE USER UserA IDENTIFIED BY sql;
```

   b. Execute the following statement to create UserB:

```
CREATE USER UserB IDENTIFIED BY sql;
```

Results

You have created two new users, UserA and UserB.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Granting Roles and Privileges (SQL) [page 1649]

Next task: Lesson 2: Grant Privileges to UserA (SQL) [page 1651]

Related Information

Creating a User (SQL Central) [page 1609]
Creating a User (SQL) [page 1610]

CREATE USER Statement

1.9.1.11.2 Lesson 2: Grant Privileges to UserA (SQL)

Grant system privileges and object-level privileges to UserA.

Prerequisites

You must have completed the previous lessons in this tutorial.

Procedure

1. In Interactive SQL, execute the following statement to grant the BACKUP DATABASE and VALIDATE ANY OBJECT system privileges to UserA:

   ```
   GRANT BACKUP DATABASE, VALIDATE ANY OBJECT TO UserA;
   ```

2. Execute the following statement to grant SELECT and UPDATE object-level privileges on the Employees table to UserA, with administration rights:

   ```
   GRANT SELECT, UPDATE ON GROUPO.EMPLOYEES TO UserA WITH GRANT OPTION;
   ```

3. This allows UserA to select from, and update, the Employees table. UserA can also grant the ability to select and update from the Employees table to other users.

Results

UserA now has the BACKUP DATABASE and VALIDATE ANY OBJECT system privileges, which are required to back up the sample database. UserA also has the SELECT and UPDATE object-level privileges on the Employees table, and can grant the same privileges to other users.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Granting Roles and Privileges (SQL) [page 1649]

Previous task: Lesson 1: Create Two Users: UserA and UserB (SQL) [page 1650]

Next task: Lesson 3: Create a Role and Grant It to UserA (SQL) [page 1652]

Related Information

System Privileges [page 1577]
Object-level Privileges [page 1573]
Granting a System Privilege (SQL Central) [page 1598]
Granting a System Privilege (SQL) [page 1599]
GRANT Statement

1.9.1.11.3 Lesson 3: Create a Role and Grant It to UserA (SQL)

Create a role that contains the privileges necessary to debug a procedure and correct any errors found during the debugging process, then grant that role to UserA.

Prerequisites

You must have completed the previous lessons in this tutorial.

Context

Make sure that UserA is granted administrative rights for the role, so that they can grant the role to, or revoke the role from, other users as necessary.

Procedure

1. In Interactive SQL, execute the following statement to create the role:

   ```
   CREATE ROLE DebugAndFix;
   ```

2. Execute the following statement to grant the required system privileges to the role:

   ```
   GRANT ALTER ANY OBJECT, DEBUG ANY PROCEDURE TO DebugAndFix;
   ```

3. Execute the following statement to grant the new role to UserA, with administrative rights:

   ```
   GRANT ROLE DebugAndFix TO UserA WITH ADMIN OPTION;
   ```

Results

The DebugAndFix role, which has the DEBUG ANY PROCEDURE and ALTER ANY OBJECT system privileges, has been created and granted to UserA, along with administrative rights.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Granting Roles and Privileges (SQL) [page 1649]

Previous task: Lesson 2: Grant Privileges to UserA (SQL) [page 1651]

Next task: Lesson 4: Convert UserA to a User-extended Role and Grant That Role to UserB (SQL) [page 1653]

Related Information

Creating a User-defined Role (SQL Central) [page 1546]
Configuring Roles and Privileges For a Role (SQL Central) [page 1561]
Granting a Role (SQL Central) [page 1563]
GRANT ROLE Statement

1.9.1.11.4 Lesson 4: Convert UserA to a User-extended Role and Grant That Role to UserB (SQL)

Convert UserA to a user-extended role and then grant that role to UserB in SQL. UserA continues to be a user who can log in to the database, in addition to being a role.

Prerequisites

You must have completed the previous lessons in this tutorial.

Context

Assume that you want UserB to have the same privileges and roles as UserA. Rather than going through the same lengthy process of granting the same roles and privileges you granted UserA to UserB, you can convert

UserA to a user-extended role and then grant that role to UserB. UserA continues to be a user who can log in to the database, in addition to being a role.

Procedure

1. In Interactive SQL, convert UserA to a user-extended role by executing the following statement:

   ```
   CREATE ROLE FOR USER UserA;
   ```

2. Grant the user-extended role UserA to UserB by executing the following statement:

   ```
   GRANT ROLE UserA TO UserB;
   ```

   This gives UserB the ability to exercise the UserA role. That is, any roles and privileges that UserA can exercise, UserB can exercise as well.

Results

UserA has been converted to a user-extended role and has been granted to UserB.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Granting Roles and Privileges (SQL) [page 1649]

Previous task: Lesson 3: Create a Role and Grant It to UserA (SQL) [page 1652]

Next task: Lesson 5: View the Roles and Privileges for UserB (SQL) [page 1655]

Related Information

User-extended Roles [page 1548]
CREATE ROLE Statement
GRANT ROLE Statement

1.9.1.11.5 Lesson 5: View the Roles and Privileges for UserB (SQL)

View the roles, system privileges, and object-level privileges for a user, including those the user inherits through membership in other roles.

Prerequisites

You must have the roles and privileges listed at the beginning of this tutorial.

Procedure

1. In Interactive SQL, execute the following statement to view the roles and privileges for UserB, including administrative rights:

```
CALL sp_displayroles ( 'UserB', 'expand_down' );
```

role_name	parent_role_name	grant_type	role_level
UserA	(NULL)	NO ADMIN	1
PUBLIC	(NULL)	NO ADMIN	1
dbo	PUBLIC	NO ADMIN	2
PUBLIC	UserA	NO ADMIN	2
DebugAndFix	UserA	ADMIN	2
BACKUP DATABASE	UserA	NO ADMIN	2
VALIDATE ANY OBJECT	UserA	NO ADMIN	2
ALTER ANY OBJECT	DebugAndFix	NO ADMIN	2
DEBUG ANY PROCEDURE	DebugAndFix	NO ADMIN	3
dbo	PUBLIC	NO ADMIN	3

Consider the role_level value as a level in a hierarchy of inheritance, where UserB is level 1:

- From rows with role_level 1, we learn that the UserA role was granted directly to UserB. We also learn that the PUBLIC role was granted to UserB directly. The PUBLIC role is automatically granted to new users.
- From rows with role_level 2, we learn that UserB inherits the dbo and PUBLIC role from UserA. UserB also inherits the DebugAndFix role that was granted to UserA, including the administrative rights on the role. Finally, UserB inherits the BACKUP DATABASE and VALIDATE ANY OBJECT system privileges that were granted to UserA.
- From rows with role_level 3, we learn that UserB inherits the ALTER ANY OBJECT and DEBUG ANY PROCEDURE system privileges from UserA, who inherited them from the DebugAndFix role. UserB

also inherits dbo again, this time from the PUBLIC role, which was automatically granted to the DebugAndFix role when it was created.

2. Execute the following statement to view the object-level privileges that UserB has.

```
CALL sp_objectpermission ( 'UserB' );
```

grantor	grantee
...	...
DBA	UserA
DBA	UserA
SYS	PUBLIC
...	...

The result set is quite long because users inherit from the PUBLIC role. The rows of interest, however, are the second and third rows in the table above because they reflect that UserB inherits SELECT and UPDATE privileges on the Employees table. We also learn that these two privileges were granted by the user DBA.

Administrative rights on object-level privileges are not inheritable, so even though DBA granted UserA the right to grant SELECT and UPDATE privileges on the Employees table, UserB does not inherit those administrative rights. The Y in the grantable column for these rows indicates that UserA has administrative rights for the privileges.

Results

You have verified the roles and privileges granted to, and inherited by, UserB.

Task overview: Tutorial: Granting Roles and Privileges (SQL) [page 1649]

Previous task: Lesson 4: Convert UserA to a User-extended Role and Grant That Role to UserB (SQL) [page 1653]

Related Information

Inheritance of Roles and Privileges [page 1617]
Role Administrators [page 1555]
Granting a System Privilege (SQL Central) [page 1598]
Granting a Role (SQL Central) [page 1563]

1.9.1.12 Upgrading from Authority-based Security to Role-based Security

You can upgrade your database from authority-based security to role-based security.

This chapter is intended for users of versions of SQL Anywhere prior to version 16.0. It explains changes when you upgrade your database to use role-based security, and what you may need to do if you have applications that execute SQL statements like the GRANT and REVOKE statements.

In this section:

In pre-16.0 databases, when you executed a system procedure that performed privileged operations, by default it executed with the privileges of the owner, typically dbo or SYS, and the SQL SECURITY clause in its definition was set to DEFINER (the owner).

1.9.1.12.1 What Happened to Authorities, Permissions, and Groups?

Previously, there were authorities, permissions, object-level permissions, and groups. SQL Anywhere 16.0 introduced a role-based security model providing roles, system privileges, object-level privileges, and user-extended roles.

SQL Anywhere 16.0 introduces a role-based security model. Whereas before you had authorities, permissions, object-level permissions, and groups, you now have roles, system privileges, object-level privileges, and user-extended roles.

> **i** Note
>
> You can use a SQL Anywhere 16.0 database server with a pre-16.0 database. When you do, full backwards compatibility is provided for that database, and its security model is not changed.

In pre-16.0 databases, **authorities** were database-level permissions. For example, a user with BACKUP authority could back up the database. Some authorities also bundled object-level permissions. For example, a user with PROFILE authority could perform database tracing tasks, which involve using system procedures that aren't otherwise available for use. You could not create new authorities, alter the permissions they were comprised of, or drop them. You could grant administrative rights (WITH GRANT), but could not limit the grant to only being an administrator.

Now, **roles** replace authorities in functionality with the added benefit that you can create new roles, alter the privileges they comprise, and drop them. Switching to roles and privileges means you have more granular control over the privileges you want to grant to a user, and an easier way to grant them to other users. You can also grant the role to a user with administrative rights only, which means the user can grant and revoke the role, but cannot exercise the underlying privileges.

In pre-16.0 databases, **permissions** allowed you to create, modify, query, use, or delete database objects such as tables, views, and users. For example, you might have SELECT permission on a table.

Now, **privileges** replace permissions in functionality, with the added benefit that there are far more privileges than permissions. For every privileged operation that can be performed on a database object, there is a grantable privilege. You can grant privileges individually to users, or grant a role to them. The term *permission* has not gone away; however, it has changed slightly. Previously, the word permission meant a grantable capability. Now, the word permission means the result of an evaluation of whether an operation can be performed. For example, you have *permission* to alter the table if you are the owner or you have the ALTER ANY TABLE system privilege.

In pre-16.0 databases, **groups** were a collection of one or more users whose authorities and permissions were determined by what is set at the group level. A user was granted group status, and then other users were granted membership in that group.

Now, the group paradigm is achieved using **user-extended roles**. If you have a user with a set of privileges that you want to grant to other users, you can extend the user to become a user-extended role, and then grant that

role to other users. The group paradigm can also be achieved by creating a **standalone role** that has the roles and privileges you want to grant to users, without being associated with a user ID.

When you upgrade a pre-16.0 database, the upgrade process automatically converts your existing authority, permission, and group hierarchy into an equivalent role, privilege, and user-extended role hierarchy. For every pre-16.0 authority, there is an equivalent **compatibility role**. These roles are easily identifiable in the database because their names start with SYS_AUTH. Compatibility roles contain the system privileges required for pre-16.0 users to perform the same operations they could perform using an authority.

Related Information

Roles [page 1527]
Privileges [page 1571]
User-extended Roles [page 1548]

1.9.1.12.2 Authorities Become Compatibility Roles

With the exception of the REMOTE DBA authority, when you upgrade a database, users that were granted authorities in pre-16.0 databases are automatically granted an equivalent compatibility role for that authority.

If a user had the ability to administer the previous authority, the user has the ability to administer the compatibility role.

Compatibility roles are not modifiable; however you can migrate them to a user-defined role, and then modify them. Migrating compatibility roles is simple, and restoring them later is also simple. When you migrate a compatibility role to a user-defined role, all users that were granted the compatibility role are automatically granted the new user-defined role. The compatibility role is automatically dropped once it has been migrated. However, you can restore compatibility roles using the CREATE ROLE statement.

Backwards compatibility for SQL statements has been provided so applications that grant or revoke authorities continue to work. However, the old syntax is deprecated and you should consider changing your applications to use the new SQL syntax for roles.

The following table shows the pre-16.0 authorities and the roles they become when a database is upgraded.

Pre-16.0 authority	Equivalent role	Description
ALL	SYS_AUTH_RESOURCE_ROLE compatibility role	Allows a user to create database objects, such as tables, views, stored procedures, and triggers.
BACKUP	SYS_AUTH_BACKUP_ROLE compatibility role	Allows a user to back up databases and transaction logs with archive or image backups by using the BACKUP DATABASE statement or dbbackup utility.

Pre-16.0 authority	Equivalent role	Description
DBA	SYS_AUTH_DBA_ROLE compatibility role (includes SYS_AUTH_SA_ROLE compatibility role and SYS_AUTH_SSO_ROLE compatibility role)	Allows users to perform all possible privileged operations. Users with the SYS_AUTH_DBA_ROLE compatibility role can create database objects and assign ownership of these objects to other user IDs, change table structures, create new user IDs, revoke permissions from users, back up the database, and so on. Of the possible privileged operations that the SYS_AUTH_DBA_ROLE compatibility role can perform, the SYS_AUTH_SA_ROLE compatibility role allows the user to perform all database administration-related activities, such as creating tables, and backing up data. Of the possible privileged operations that the SYS_AUTH_DBA_ROLE compatibility role can perform, the SYS_AUTH_SSO_ROLE compatibility role allows the user to perform the security and access-related administration activities, such as creating users, and granting privileges on objects.
PROFILE	SYS_AUTH_PROFILE_ROLE compatibility role	Allows a user to perform profiling, tracing, and diagnostic operations.
READCLIENTFILE	SYS_AUTH_READCLIENTFILE_ROLE compatibility role	Allows a user to read files on the client computer, for example when loading data from a file on a client computer.
READFILE	SYS_AUTH_READFILE_ROLE compatibility role	Allows a user to use the OPENSTRING clause in a SELECT statement to read a file.

Pre-16.0 authority	Equivalent role	Description
REMOTE DBA	SYS_RUN_REPLICATION_ROLE **system** role	Allows a SQL Remote user to perform replication activities by using the dbremote utility, and a MobiLink user to perform synchronization activities by using the dbmlsync utility. It does not allow administration of replication, however. There is also a new system role, SYS_REPLICATION_ADMIN_ROLE you can grant to replication administrators. In pre-16.0 databases, replication administrators needed DBA authority to perform administrative tasks. The SYS_REPLICATION_ADMIN_ROLE system role encompasses the privileges needed to perform those administrative tasks.
RESOURCE	SYS_AUTH_RESOURCE_ROLE compatibility role	Allows a user to create database objects, such as tables, views, stored procedures, and triggers.
VALIDATE	SYS_AUTH_VALIDATE_ROLE compatibility role	Allows a user to perform database, table, index, and checksum validation by using the VALIDATE statement or dbvalid utility.
WRITECLIENTFILE	SYS_AUTH_WRITECLIENTFILE_ROLE compatibility role	Allows a user to write to files on a client computer, for example when using the UNLOAD TABLE statement to write data to a client computer.
WRITEFILE	SYS_AUTH_WRITEFILE_ROLE compatibility role	Allows a user to execute the xp_write_file system procedure.

Related Information

Roles [page 1527]
Replication-related System Roles [page 1530]
Changes to the GRANT Statement Syntax [page 1664]
Changes to the REVOKE Statement Syntax [page 1669]
Changes in Inheritance Behavior for Some Authorities That Became Roles [page 1675]
CREATE ROLE Statement

1.9.1.12.3 Permissions Become Privileges

In version 16.0, permissions became privileges.

In pre-16.0 databases, there were object-level permissions such as ALTER and SELECT for tables and views, and so on. While statements that grant or revoke these permissions still work, these permissions are now referred to as privileges but retain the same name. In addition to object-level privileges, there is a grantable system privilege for every operation that requires authorization to perform. When you upgrade your database, users that had permissions are automatically updated to have the equivalent privileges they need to perform the tasks they could perform before.

Related Information

Privileges [page 1571]

1.9.1.12.4 Groups Are Now Achieved Using Roles

During the upgrade of a pre-16.0 database, each group is converted either to a user-extended role, or to a standalone role of the same name.

Members of the original group are automatically granted the new role and all of its underlying privileges. Authorities and object-level permissions that were granted to the original group are converted to their equivalent roles and system privileges and granted to the user-extended role.

If an authority was inheritable, the compatibility role will be inherited by grantees of the new user-extended role. If the authority was non-inheritable, the grantees of the user-extended role do not inherit the compatibility role.

The following table shows the system users and groups and the roles they are converted to.

Pre-16.0 group	Role	Description
dbo	dbo	This role owns many system stored procedures, views, and tables used for UltraLite and MobiLink.
DIAGNOSTICS	DIAGNOSTICS	This role owns the diagnostic tables and views, and can perform operations on them.
PUBLIC	PUBLIC	This role has SELECT permission on the system tables. Any new user ID is automatically granted the PUBLIC role.
SA_DEBUG	SA_DEBUG	This role allows users to use the SQL Anywhere Debugger.

Pre-16.0 group	Role	Description
SYS	SYS	This role owns the system tables and views (catalog) for the database, and can perform operations on them.
SYS_SPATIAL_ADMIN_ROLE	SYS_SPATIAL_ADMIN_ROLE	This role allows users to create, alter, or drop spatial objects.

The SQL syntax for creating and administering groups has changed, although the previous GRANT and REVOKE syntax is still supported.

Related Information

Groups [page 1615]
Roles [page 1527]
Changes to the REVOKE Statement Syntax [page 1669]
Changes to the GRANT Statement Syntax [page 1664]

1.9.1.12.5 Change to Concept of a Database Super-user (DBA Authority)

In pre-16.0 databases, you could create a database super-user by granting them DBA authority.

Users with DBA authority could perform any privileged task in the database. When you upgrade your database, any users that had DBA authority get the SYS_AUTH_DBA_ROLE compatibility role, and automatically receive exercise and administration rights for all roles and privileges that are present at the time of upgrade.

Also, when you create a new role and don't specify an administrator at creation time, users with the MANAGE ROLES system privilege (global administrators) can administer the role.

However, if you create a new role and assign administrators as part of role creation, administration is then limited to the grantees who were given administration rights. Therefore, if you want your super-user to have administrative rights for the new role, you must grant it explicitly.

Related Information

Super-users [page 1558]
Role Administrators [page 1555]

1.9.1.12.6 Changes to the GRANT Statement Syntax

If you have applications that use the pre-16.0 GRANT statement syntax for authorities, permissions, and groups, you should modify them to use the updated GRANT ROLE and GRANT syntax for roles and privileges.

The table below shows you what the statements should be changed to. Use of the old GRANT syntax for authorities, permissions, and groups is supported but deprecated.

> **i Note**
>
> In pre-16.0 databases, DBA, BACKUP, RESOURCE, and VALIDATE authorities were non-inheritable in the case where they were assigned to a user acting as a group. Members of that group did not inherit the capabilities of the authority.
>
> When you upgrade a pre-16.0 database, the equivalent role is automatically granted (for example, SYS_AUTH_BACKUP_ROLE is granted instead of BACKUP authority), and the WITH NO SYSTEM PRIVILEGE INHERITANCE clause is specified to ensure that inheritance behavior remains consistent with previous releases. Members of the group do not inherit the privileges that the role provides.
>
> Likewise, if you continue to use the deprecated grant syntax to grant DBA, BACKUP, RESOURCE, and VALIDATE, the old behavior of non-inheritance is maintained. That is, the equivalent role is granted and the WITH NO SYSTEM PRIVILEGE INHERITANCE clause is specified. This is done automatically.

In pre-16.0 databases, users that were granted DBA authorities automatically could grant them to others. The WITH ADMIN OPTION clause in the new syntax recommended below ensures that administration rights behavior remains consistent with previous releases.

Deprecated syntax	New syntax
GRANT CONNECT TO `user-name` `[` *IDENTIFIED BY* `]` `password`	*CREATE USER* `user-name` `[` *IDENTIFIED BY* `password` `]`
GRANT GROUP TO `userid`	*CREATE OR REPLACE ROLE FOR USER* `userid`
GRANT MEMBERSHIP IN GROUP `groupname [,...]` *TO* `grantee [,...]`	*GRANT ROLE* `groupname [,...]` *TO* `grantee [,...]`
GRANT DBA TO `grantee [,...]`	*GRANT ROLE SYS_AUTH_DBA_ROLE TO* `grantee [,...]` *WITH ADMIN OPTION* *WITH NO SYSTEM PRIVILEGE INHERITANCE*
GRANT REMOTE DBA TO `grantee [,...]`	*GRANT ROLE SYS_RUN_REPLICATION_ROLE TO* `grantee [,...]` *WITH NO ADMIN OPTION* *WITH NO SYSTEM PRIVILEGE INHERITANCE*

Deprecated syntax	New syntax
GRANT BACKUP TO `grantee [,...]`	*GRANT ROLE SYS_AUTH_BACKUP_ROLE TO* `grantee [,...]` *WITH NO ADMIN OPTION* *WITH NO SYSTEM PRIVILEGE INHERITANCE*
GRANT RESOURCE TO `grantee [,...]`	*GRANT ROLE SYS_AUTH_RESOURCE_ROLE TO* `grantee [,...]` *WITH NO ADMIN OPTION* *WITH NO SYSTEM PRIVILEGE INHERITANCE*
GRANT VALIDATE TO `grantee [,...]`	*GRANT ROLE SYS_AUTH_VALIDATE_ROLE TO* `grantee [,...]` *WITH NO ADMIN OPTION* *WITH NO SYSTEM PRIVILEGE INHERITANCE*
GRANT PROFILE TO `grantee [,...]`	*GRANT ROLE SYS_AUTH_PROFILE_ROLE TO* `grantee [,...]` *WITH NO ADMIN OPTION*
GRANT READCLIENTFILE TO `grantee [,...]`	*GRANT ROLE SYS_AUTH_READCLIENTFILE_ROLE TO* `grantee [,...]` *WITH NO ADMIN OPTION*
GRANT READFILE TO `grantee [,...]`	*GRANT ROLE SYS_AUTH_READFILE_ROLE TO* `grantee [,...]` *WITH NO ADMIN OPTION*
GRANT WRITECLIENTFILE TO `grantee [,...]`	*GRANT ROLE SYS_AUTH_WRITECLIENTFILE_ROLE TO* `grantee [,...]` *WITH NO ADMIN OPTION*
GRANT WRITEFILE TO `grantee [,...]`	*GRANT ROLE SYS_AUTH_WRITEFILE_ROLE TO* `grantee [,...]` *WITH NO ADMIN OPTION*
GRANT PUBLISH TO `grantee`	No change. However, you can also set the new PUBLIC option, db_publisher, and provide the user_id of the grantee instead of the user name of the grantee: *SET OPTION PUBLIC.db_publisher=USER_ID('grantee')*

Deprecated syntax	New syntax
GRANT `permission [,...]` *ON* `[ owner.]object-name` *TO* `grantee [,...]` `permission :` *ALL* `[` *PRIVILEGES* `]` `|` *ALTER* `|` *DELETE* `|` *INSERT* `|` *REFERENCES* `[` *(* `column-name, ...)` `]` `|` *SELECT* `[` *(* `column-name, ... )` `]` `|` *UPDATE* `[` *(* `column-name, ... )` `]`	No change.
GRANT EXECUTE ON `[owner.]{ proc	user-def-funct }` *TO* `grantee [,...]`	No change.
GRANT INTEGRATED LOGIN TO `user-profile-name [,...]` *AS USER* `user`	No change.
GRANT KERBEROS LOGIN TO `userid [,...]` *AS USER* `user`	No change.
GRANT CREATE ON `dbspacename [,...]` *TO* `grantee [,...]`	No change.

In this section:

GRANT Statement (Authorities and Groups) (Deprecated) [page 1667]
Use the new GRANT statement for granting system privileges (previously permissions), and GRANT ROLE statement for granting roles (previously authorities and groups).

Related Information

Compatibility Roles [page 1532]
Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
GRANT Statement
GRANT ROLE Statement

1.9.1.12.6.1 GRANT Statement (Authorities and Groups) (Deprecated)

Use the new GRANT statement for granting system privileges (previously permissions), and GRANT ROLE statement for granting roles (previously authorities and groups).

Syntax 1 - Grant Authorities

```
GRANT authority, ...
 TO userid, ...
```

```
authority :
BACKUP
| DBA
| PROFILE
| READCLIENTFILE
| READFILE
| [ RESOURCE | ALL ]
| VALIDATE
| WRITECLIENTFILE
| WRITEFILE
```

Syntax 2 - Grant Group Status to a User, or Grant Membership in a Group to a User

```
GRANT { GROUP | MEMBERSHIP IN GROUP userid, ... }
 TO userid, ...
```

Parameters

authority

> **i** Note
>
> When you grant authorities using the old deprecated GRANT syntax, the database server converts the authority to its equivalent compatibility roles. No capabilities are lost for the user being granted the compatibility role equivalent.

BACKUP authority

Converted to granting the SYS_AUTH_BACKUP_ROLE compatibility role.

DBA authority

Converted to granting the SYS_AUTH_DBA_ROLE, SYS_AUTH_SA_ROLE, and SYS_AUTH_SSO_ROLE compatibility roles.

PROFILE authority

Converted to granting the SYS_AUTH_PROFILE_ROLE compatibility role.

READCLIENTFILE authority

Converted to granting the SYS_AUTH_READCLIENTFILE_ROLE compatibility role.

READFILE authority

Converted to granting the SYS_AUTH_READFILE_ROLE compatibility role.

RESOURCE or ALL authority

Converted to granting the SYS_AUTH_RESOURCE_ROLE compatibility role.

VALIDATE authority

Converted to granting the SYS_AUTH_VALIDATE_ROLE compatibility role.

WRITECLIENTFILE authority

Converted to granting the SYS_AUTH_WRITECLIENTFILE_ROLE compatibility role.

WRITEFILE authority

Converted to granting the SYS_AUTH_WRITEFILE_ROLE compatibility role.

GROUP clause

Granting this now converts the user to a user-extended role that can be granted to others.

MEMBERSHIP IN GROUP clause

Granting this gives the grantee all the roles and privileges of the user-extended role (group) they are being granted.

Remarks

The syntax for granting authorities is deprecated but supported for backwards compatibility. Granting an authority results in granting its equivalent compatibility role that is provided in version 16.0 and up. However, if you have migrated your compatibility roles, granting an authority using the deprecated syntax will fail. For example, if you migrate your SYS_AUTH_BACKUP_ROLE compatibility role to a user-defined role (which drops the compatibility role when it finishes the migration), and then execute `GRANT BACKUP TO UserA`, the statement fails because the SYS_AUTH_BACKUP_ROLE compatibility role no longer exists in the database. You can resolve this issue by restoring the compatibility role (CREATE ROLE SYS_AUTH_BACKUP_ROLE), or by changing your applications to use the new syntax and reference the new user-defined role you created by migrating.

> **i** Note
>
> In pre-16.0 databases, DBA, REMOTE DBA, BACKUP, RESOURCE, and VALIDATE authorities were non-inheritable in the case where they were assigned to a user acting as a group. Members of that group did not inherit the capabilities of the authority.
>
> When you upgrade a pre-16.0 database, the equivalent role is automatically granted (for example, SYS_AUTH_BACKUP_ROLE is granted instead of BACKUP authority), and the WITH NO SYSTEM PRIVILEGE INHERITANCE clause is specified to ensure that inheritance behavior remains consistent with previous releases. Members of the group do not inherit the privileges that the role provides.

Likewise, if you continue to use the deprecated grant syntax to grant DBA, REMOTE DBA, BACKUP, RESOURCE, and VALIDATE, the old behavior of non-inheritance is maintained. That is, the equivalent role is granted and the WITH NO SYSTEM PRIVILEGE INHERITANCE clause is specified. This is done automatically.

Privileges

You must have administrative rights for the equivalent role you are granting. For example, if you are granting the ability to back up databases, and are using the GRANT BACKUP syntax, you must have administration rights on the SYS_AUTH_BACKUP_ROLE compatibility role.

Related Information

Compatibility Roles [page 1532]
Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
Viewing the Roles and Privileges for a User or Role (SQL) [page 1613]
GRANT Statement
GRANT ROLE Statement
GRANT statement (SQL Anywhere 12.0.1)

1.9.1.12.7 Changes to the REVOKE Statement Syntax

If you have applications that use the pre-16.0 REVOKE statement syntax for authorities, permissions, and groups, you should modify them to use the updated REVOKE ROLE and REVOKE syntax for roles and privileges.

The table below shows you what the statements should be changed to. Use of the old REVOKE syntax for authorities, permissions, and groups is supported but deprecated.

Pre-16.0 syntax	New syntax
REVOKE CONNECT FROM `userid`	No change.
REVOKE GROUP FROM `userid`	*DROP ROLE FROM USER* `rolename` *WITH REVOKE*
REVOKE MEMBERSHIP IN GROUP `groupname [,...]` *FROM* `grantee [,...]`	*REVOKE ROLE* `groupname [,...]` *FROM* `grantee [,...]`

Pre-16.0 syntax	New syntax
REVOKE `authority` *FROM* `grantee [,...]` `authority :` *DBA* `|` *REMOTE DBA* `|` *BACKUP* `|` *RESOURCE* `|` *VALIDATE* `|` *PROFILE* `|` *READCLIENTFILE* `|` *READFILE* `|` *WRITECLIENTFILE* `|` *WRITEFILE*	*REVOKE ROLE* `rolename [,...]` *FROM* `userid [,...]` `role :` *SYS_AUTH_DBA_ROLE* `|` *SYS_RUN_REPLICATION_ROLE* `|` *SYS_AUTH_BACKUP_ROLE* `|` *SYS_AUTH_RESOURCE_ROLE* `|` *SYS_AUTH_VALIDATE_ROLE* `|` *SYS_AUTH_PROFILE_ROLE* `|` *SYS_AUTH_READCLIENTFILE_ROLE* `|` *SYS_AUTH_READFILE_ROLE* `|` *SYS_AUTH_WRITECLIENTFILE_ROLE* `|` *SYS_AUTH_WRITEFILE_ROLE*
REVOKE PUBLISH FROM `grantee`	No change. However, you can also unset the new PUBLIC option, db_publisher, as follows: *SET OPTION PUBLIC.db_publisher=*`grantee`
REVOKE `permission [,...]` *ON* `[ owner.]object-name` *FROM* `grantee [,...]` `permission :` *ALL* `[` *PRIVILEGES* `]` `|` *ALTER* `|` *DELETE* `|` *INSERT* `|` *REFERENCES* `[ ( column-name, ...) ]` `|` *SELECT* `[ ( column-name, ... ) ]` `|` *UPDATE* `[ ( column-name, ... ) ]`	No change, except to naming convention. Object-level permissions are now object-level privileges.
REVOKE EXECUTE ON `[ owner.]` `{ procedure-name	user-defined-function }` *FROM* `grantee [,...]`	No change.
REVOKE INTEGRATED LOGIN FROM `user-profile-name [,...]`	No change.
REVOKE KERBEROS LOGIN FROM `userid [,...]` *AS USER* `user`	No change.
REVOKE CREATE ON `dbspacename [,...]` *FROM* `grantee [,...]`	No change.

To review the SQL Anywhere 12.0.1 REVOKE statement syntax, see REVOKE statement.

In this section:

Related Information

Compatibility Roles [page 1532]
Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
Viewing the Roles and Privileges for a User or Role (SQL) [page 1613]
REVOKE Statement
REVOKE ROLE Statement

1.9.1.12.7.1 REVOKE Statement (Authorities and Groups) (Deprecated)

Use the new REVOKE statement for revoking system privileges (previously permissions), and REVOKE ROLE statement for revoking roles (previously authorities and groups).

> **i** Note
>
> When you revoke authorities using the deprecated REVOKE syntax, the database server converts the revoke to the equivalent compatibility role instead.

Syntax

```
REVOKE authority, ... FROM userid, ...
```

```
authority :
[ ALL | RESOURCE ]
| BACKUP
| CONNECT
| DBA
| GROUP
| MEMBERSHIP IN GROUP userid, ...
| PROFILE
| READCLIENTFILE
| READFILE
| REMOTE DBA
| VALIDATE
| WRITECLIENTFILE
| WRITEFILE
```

Parameters

authority

> **i** Note
>
> When you revoke authorities using the old deprecated REVOKE syntax, the database server converts the authority to its equivalent compatibility roles.

ALL or RESOURCE authority

To revoke ALL or RESOURCE authority from a user, revoke the SYS_AUTH_RESOURCE_ROLE compatibility role.

BACKUP authority

To revoke BACKUP authority from a user, revoke the SYS_AUTH_BACKUP_ROLE compatibility role.

CONNECT authority

To revoke CONNECT authority from a user (delete a user), use DROP USER `userid`.

DBA authority

To revoke DBA authority from a user, revoke the SYS_AUTH_DBA_ROLE, SYS_AUTH_SA_ROLE, and SYS_AUTH_SSO_ROLE compatibility roles.

GROUP authority

To convert a user-extended role (group) back to a regular user so they are no longer a group that users can be added to, use DROP ROLE FROM USER `userid`.

MEMBERSHIP IN GROUP clause

To revoke MEMBERSHIP IN GROUP, use REVOKE ROLE `groupname` [,...] FROM `userid`.

PROFILE authority

To revoke PROFILE authority from a user, revoke the SYS_AUTH_PROFILE_ROLE compatibility role.

READCLIENTFILE authority

To revoke READCLIENTFILE authority from a user, revoke the SYS_AUTH_READCLIENTFILE_ROLE compatibility role.

READFILE authority

To revoke READFILE authority from a user, revoke the SYS_AUTH_READFILE_ROLE compatibility role.

REMOTE DBA authority

To revoke REMOTE DBA authority from a user, revoke the SYS_RUN_REPLICATION_ROLE system role.

VALIDATE authority

To revoke VALIDATE authority from a user, revoke the SYS_AUTH_VALIDATE_ROLE compatibility role.

WRITECLIENTFILE authority

To revoke WRITECLIENTFILE authority from a user, revoke the SYS_AUTH_WRITECLIENTFILE_ROLE compatibility role.

WRITEFILE authority

To revoke WRITEFILE authority from a user, revoke the SYS_AUTH_WRITEFILE_ROLE compatibility role.

Remarks

REVOKE GROUP automatically revokes MEMBERSHIP IN GROUP from all members of the group.

When you add a user to a group, the user inherits all the privileges assigned to that group. You are not allowed to revoke a subset of the privileges that a user inherits as a member of a group because you can only revoke privileges that are explicitly given by a GRANT statement. If you require different privileges for different users, create different groups with the appropriate privileges, or explicitly grant each user the privileges they require.

If you revoke a privilege for a user and they also had WITH GRANT OPTION for that privilege, then everyone who that user granted the privilege to also has their privilege revoked. For example, suppose you granted UserA SELECT...WITH GRANT OPTION privileges on a table and UserA then grants the SELECT privilege on the table to UserB. If you revoke the SELECT privilege from UserA, it is revoked from UserB as well.

To review the SQL Anywhere 12.0.1 REVOKE statement syntax, see REVOKE statement.

Privileges

You must have administrative rights for the equivalent role you are revoking. For example, if you are revoking the ability to back up databases, and are using the REVOKE BACKUP syntax, you must have administration rights on the SYS_AUTH_BACKUP_ROLE compatibility role.

Related Information

Compatibility Roles [page 1532]
Viewing the Roles and Privileges for a User or Role (SQL Central) [page 1612]
REVOKE ROLE Statement

1.9.1.12.8 Changes to REMOTE DBA

In pre-16.0 databases, REMOTE DBA authority allowed a user to perform replication and synchronization operations using dbremote and dbmlsync.

The REMOTE DBA authority has been replaced by the SYS_RUN_REPLICATION_ROLE system role. Change your applications to grant this role, instead of REMOTE DBA. The GRANT REMOTE DBA statement syntax is still supported but deprecated.

Another replication-related role has also been introduced: the SYS_REPLICATION_ADMIN_ROLE system role. This role allows user to administer replication.

In this section:

GRANT REMOTE DBA Statement (Deprecated) [page 1674]
The GRANT REMOTE DBA grants the SYS_RUN_REPLICATION_ROLE system role, which is required to run the dbremote utility for replication and the dbmlsync utility for synchronization.

Related Information

Replication-related System Roles [page 1530]
GRANT ROLE Statement
REVOKE ROLE Statement

1.9.1.12.8.1 GRANT REMOTE DBA Statement (Deprecated)

The GRANT REMOTE DBA grants the SYS_RUN_REPLICATION_ROLE system role, which is required to run the dbremote utility for replication and the dbmlsync utility for synchronization.

Syntax

```
GRANT REMOTE DBA
TO userid, ...
[ IDENTIFIED BY password ]
```

Parameters

IDENTIFIED BY clause

The IDENTIFIED BY clause is optional for this statement. If included, the password for the user is changed.

Remarks

Executing this statement grants the SYS_RUN_REPLICATION_ROLE system role. The REMOTE DBA authority is deprecated.

If you use this statement in a procedure, do not specify the password as a string literal because the definition of the procedure is visible in the SYSPROCEDURE system view. For security purposes, specify the password using a variable that is declared outside of the procedure definition.

Related Information

Replication-related System Roles [page 1530]
GRANT ROLE Statement

1.9.1.12.9 Changes in Inheritance Behavior for Some Authorities That Became Roles

In pre-16.0 databases, if you granted the DBA, REMOTE DBA, BACKUP, RESOURCE AND VALIDATE authorities to a group, the underlying permissions were *not* inherited by members of the group.

Now, however, the default behavior when granting *any* role (including SYS_AUTH_DBA_ROLE, SYS_RUN_REPLICATION_ROLE, SYS_AUTH_BACKUP_ROLE, SYS_AUTH_RESOURCE_ROLE, and SYS_AUTH_VALIDATE_ROLE) to a user-defined role is to allow those who have been granted the user-defined role to inherit the underlying system privileges of these roles.

Suppose you have a user, userA. You grant userA the ALTER ANY OBJECT system privilege. You then decide to extend userA to become a role, and then grant userA to userB. Now you want to grant the SYS_AUTH_DBA_ROLE compatibility role to userA, but you don't want userB to inherit all the privileges that the SYS_AUTH_DBA_ROLE compatibility role gives. You would therefore grant the SYS_AUTH_DBA_ROLE compatibility role as follows:

```
GRANT ROLE SYS_AUTH_DBA_ROLE TO userA WITH NO SYSTEM PRIVILEGE INHERITANCE;
```

In this scenario, userB inherits only the ALTER ANY OBJECT system privilege from userA.

To retain the non-inheritance behavior of these roles during upgrade, aa WITH NO SYSTEM PRIVILEGE INHERITANCE clause is supported in the GRANT ROLE statement. Likewise, if you have applications that you are changing to use the new GRANT syntax, you must specify this clause as well. This clause is only for use with these specific roles.

Related Information

Compatibility Roles [page 1532]
Granting a Compatibility Role (SQL Central) [page 1538]
Granting a Compatibility Role (SQL) [page 1539]
GRANT ROLE Statement

1.9.1.12.10 Changes in Administering the Database Publisher

In pre-16.0 databases, the database publisher was controlled by granting the PUBLISH authority by using the GRANT PUBLISH and REVOKE PUBLISH statements.

The current publisher could be determined by querying the CURRENT PUBLISHER special value.

The PUBLISH authority has been replaced by the PUBLIC.db_publisher database option, which requires the SET ANY SYSTEM OPTION system privilege to be set. Changing the publisher can be achieved by changing the database option, but for backwards compatibility, you can still change it using GRANT PUBLISH and REVOKE PUBLISH. You can also still query the CURRENT PUBLISHER to find out the current publisher.

Related Information

db_publisher Option [page 720]
GRANT PUBLISH Statement [SQL Remote]
REVOKE PUBLISH Statement [SQL Remote]

1.9.1.12.11 Changes For System Procedures That Perform Privileged Operations

In pre-16.0 databases, when you executed a system procedure that performed privileged operations, by default it executed with the privileges of the owner, typically dbo or SYS, and the SQL SECURITY clause in its definition was set to DEFINER (the owner).

In version 16.0, the SQL SECURITY clause for these system procedures is set to INVOKER by default. That is, the privileged operations will execute with the privileges of the invoker, instead of the definer.

Now, when you create or upgrade a database, you can specify whether system procedures that perform privileged operations will execute with the privileges of the owner or the invoker. The CREATE DATABASE statement, ALTER DATABASE UPGRADE statement, Initialization utility (dbinit), and the Upgrade utility (dbupgrad), have been enhanced to allow this choice.

For new databases, if nothing is specified, invoker is chosen for these system procedures. When upgrading, if nothing is specified, the default is whatever is already in place for the database being upgraded.

Related Information

Procedures and Functions Running with Owner or Invoker Privileges
ALTER DATABASE Statement
CREATE DATABASE Statement
Initialization Utility (dbinit) [page 1166]
Upgrade Utility (dbupgrad) [page 1254]

1.9.2 Data Security

Databases may contain proprietary, confidential, or private information, making it important to ensure that the database and the data in it are designed for security.

Several features are included to assist in building a secure environment for your data:

System privileges and roles

These features control who has access to a database.

Discretionary access control features

These features control the actions a user can perform while connected to a database.

Auditing

This feature helps you maintain a record of actions on the database.

Database server options

These features let you control who can perform administrative operations (for example, loading databases). These options are set when you start the database server.

Views and stored procedures

These features allow you to specify the data a user can access and the operations a user can execute.

Database, table, and arbitrary data encryption

You can choose to secure your database either with simple obfuscation, or with strong encryption. Simple obfuscation is not secure against skilled and determined attempts to gain access to the data. Strong encryption renders the database completely inaccessible without an encryption key.

Table encryption features allow you to encrypt individual tables, instead of encrypting the entire database.

Arbitrary data encryption allows you to encrypt blocks of data, such as messages, with a public or private key and decrypt it with the corresponding private or public key. You can also sign data with a private key and verify it with the corresponding public key.

Transport Layer Security (TLS)

You can use transport layer security to authenticate and encrypt communications between client applications and the database server. Transport layer security uses RSA encryption technology.

Use TLS encryption when the computer that runs your database server also runs other processes, which could access the contents of your client/server communications.

Secure features

You can disable features for all databases running on a database server.

SELinux support

SELinux policies control an application's access to system resources. A policy that secures it on Red Hat Enterprise Linux 5 is included.

For information about compiling and installing the SQL Anywhere SELinux policy, see *$SQLANY17*`/selinux/readme`.

In this section:

Secured Features [page 1688]
Features can be made inaccessible to databases running on a database server. These features are secured from use.

Database Activity Audits [page 1692]
Auditing tracks all of the activity performed on a database.

Database Encryption and Decryption [page 1697]
Make it more difficult for someone to decipher the data in your database.

Message Encryption Using Public and Private Keys [page 1723]
Encrypt, sign, and verify messages with an RSA key pair.

Related Information

Administration Tools Security (Interactive SQL) [page 1094]
Users [page 1608]
Security: Control Privileges from the Command Line [page 1632]
Table Encryption [page 1706]
Transport Layer Security [page 1727]

1.9.2.1 Security: Use Views and Procedures to Limit Data Users Can Access

For databases that require a high level of security, granting access directly to tables has limitations.

Any privilege granted to a user on a table applies to the whole table. There are many cases when users' privileges need to be shaped more precisely than on a table-by-table basis. For example:

- It is not desirable to give access to personal or sensitive information stored in an employee table to users who need access to other parts of the table.
- You may want to give sales representatives update privileges on a table containing descriptions of their sales calls, but limit such privileges to their own calls.

While views restrict data access, procedures restrict the actions a user may take. A user can have EXECUTE privilege on a procedure without having any privileges on the table or tables on which the procedure acts.

For strict security, you can disallow all access to the underlying tables, and grant privileges to users or groups of users to execute certain stored procedures. This approach strictly defines how data in the database can be modified.

Example

The Sales manager needs access to information in the database concerning employees in the department. However, there is no reason for the manager to have access to information about employees in other departments.

This example describes how to create a user ID for the sales manager, create views that provide the information she needs, and grant the appropriate privileges to the sales manager user ID.

1. Connect and create the new user ID:

```
CONNECT DBA IDENTIFIED by sql;
CREATE USER SalesManager
IDENTIFIED BY sales;
```

2. Define a view that only looks at sales employees as follows:

```
CREATE VIEW EmployeeSales AS
  SELECT EmployeeID, GivenName, Surname
  FROM Employees
  WHERE DepartmentID = 200;
```

   The table reference could be qualified with the owner to avoid an ambiguous reference to an identically named table.

3. Give SalesManager privilege to look at the view:

```
GRANT SELECT ON EmployeeSales TO SalesManager;
```

   You use exactly the same statement to grant privilege on views and on tables.

The next example creates a view which allows the Sales Manager to look at a summary of sales orders. This view requires information from more than one table for its definition:

1. Create the view.

```
CREATE VIEW OrderSummary AS
  SELECT OrderDate, Region, SalesRepresentative, CompanyName
  FROM SalesOrders
    KEY JOIN Customers;
```

2. Grant privilege for the Sales Manager to examine this view.

```
GRANT SELECT
ON OrderSummary
TO SalesManager;
```

3. To check that the process has worked properly, connect to the SalesManager user ID and look at the views you created:

```
CONNECT SalesManager
IDENTIFIED BY sales;
SELECT *
FROM DBA.EmployeeSales;
SELECT *
FROM DBA.OrderSummary;
```

No privileges have been granted to the Sales Manager to look at the underlying tables. The following statements produce privilege errors.

```
SELECT * FROM GROUPO.Employees;
SELECT * FROM GROUPO.SalesOrders;
```

In this section:

Security: Procedures and Triggers [page 1680]

Procedures and triggers provide security by allowing users limited access to data in tables that they cannot directly examine or modify.

1.9.2.1.1 Security: Procedures and Triggers

Procedures and triggers provide security by allowing users limited access to data in tables that they cannot directly examine or modify.

Triggers, for example, execute under the table privileges of the owner of the associated table, but any user with privileges to insert, update or delete rows in the table can fire them. Similarly, procedures (including user-defined functions) execute with privileges of the procedure owner, but any user granted privileges can call them. Procedures and triggers can (and usually do) have different privileges than the user ID that invoked them.

Procedures

In high-security systems, consider disallowing access to all base tables, and instead permit access to data and tasks through procedures. To restrict access using procedures:

1. Create a role for each set of authorized tasks to be performed and grant the role the applicable system privileges.
2. Grant each of these roles to a single common role.
3. Grant EXECUTE privileges on the procedure for performing the authorized tasks to the applicable role.
4. When a new user is created, grant only the roles required for the tasks the user will perform.

Related Information

sp_proc_priv System Procedure
ALTER DATABASE Statement
CREATE DATABASE Statement
CREATE PROCEDURE Statement
Initialization Utility (dbinit) [page 1166]
Upgrade Utility (dbupgrad) [page 1254]

1.9.2.2 General Security Tips

There are many actions you can take to improve the security of your data.

Choose Passwords Carefully

Never deploy databases with short or easy-to-guess passwords.

Restrict Use of Super-Users

Avoid creating and granting roles that possess all privileges, roles, and administration rights (that is, avoid super-users). Instead, create roles with logical groupings of privileges and rights, and then grant those roles judiciously. If you do create super-users, consider using them only when absolutely necessary, and store their passwords in a secure location, such as a safe, so that you can retrieve them when needed.

Consider giving your database administrators two user IDs: one with all privileges and one with limited privileges, so they can connect to the one with all privileges only when necessary.

Use Secured Database Features

The -sf database server option lets you enable and disable features for all databases running on a database server. The features you can disable include the use of external stored procedures, Java, remote data access, and the ability to change the request log settings.

Disabling Database Features

The -sf database server option specifies a list of features that are disabled for databases running on the database server so they are not available to client applications or stored procedures, triggers, or events defined within the databases. This can be useful when you are starting a database that is not your own that may attempt unwanted operating system interactions such as file transfers between the database client and server.

Drop External System Functions

The following external functions present possible security risks:

- xp_cmdshell system procedure
- xp_startmail system procedure,
- xp_startsmtp system procedure
- xp_sendmail system procedure
- xp_stopmail system procedure
- xp_stopsmtp system procedure

The email system procedures allow users to have the database server send email composed by the user. Malicious users could use either the email or command shell procedures to perform operating-system tasks with permissions other than those they have been given by the operating system. In a security-conscious environment, you should drop these functions.

Protect Your Database Files

Protect the database file, log files, and dbspace files from unauthorized access. Do not store them on a shared directory or volume.

Protect Your Database Software

Protect your SQL Anywhere software from unauthorized access. Only give users access to the applications, DLLs, and other resources they require.

Run the Database Server as a Service or a Daemon

Prevent unauthorized users from shutting down or gaining access to the database or log files, by running the database server as a Windows service. On Unix, running the server as a daemon serves a similar purpose.

Set the SATMP Environment Variable to a Unique Directory (Unix)

Make the database server secure on Unix platforms, by setting SATMP to a unique directory, and make the directory read, write, and execute protected against all other users. Doing so forces all other connections to use TCP/IP, which is more secure than the shared memory connection.

The shared memory buffers that are used between the client and server are removed from the directory tree before any actual data is sent between the two sides. Another process cannot see any of the communication data because the shared memory buffer/file is hidden.

Strongly Encrypt Your Database

Strongly encrypting your database makes it completely inaccessible without the key. You cannot open the database or view the database or transaction log files using any other means.

Adjust the -xs Server Option HTTP MaxRequestSize Protocol Option

When using the -xs HTTP server option, set the MaxRequestSize option to prevent very large files from being sent to the database server, which can tie up server resources and bandwidth.

Set the HTTP Timeout Options

When using the -xs HTTP server option, set the Timeout (TO) and KeepaliveTimeout (KTO) options. Timeout indicates how long HTTP connections remain active before a request is received, and KeepaliveTimeout indicates how long keep-alive connections remain open between requests.

Set the request_timeout Option to Ensure That No One Request Can Consume Server Resources Indefinitely

Set the request_timeout database option to ensure that no single request can consume server resources indefinitely.

Secure the Disk Sandbox Feature

If disk sandboxing is turned on, then database file operations are restricted to the directory where the main database file is located and any subdirectories of this directory.

Starting and Stopping Databases

When using a personal database server (dbeng17), by default any user can start an extra database on a running server.

By default, network database servers (dbsrv17) require the SERVER OPERATOR system privilege to start a database on a running database server. The -gd database option allows you to limit access to this option to users with a certain level of privileges in the database to which they are already connected.

Creating and Deleting Databases

When running a personal database server (dbeng17), by default any user can use the CREATE DATABASE or DROP DATABASE statements to create or delete a database file.

By default, network database servers (dbsrv17) require the SERVER OPERATOR system privilege to create or delete databases. The -gu option allows you to limit access to this option to users with a certain level of privilege in the database to which they are connected.

Stopping the Server

The dbstop utility stops a database server. It is useful in batch files, or in other cases where stopping the server interactively (by clicking *Shut Down* on the database server messages window) is impractical.

By default on personal database servers (), any user can run dbstop to shut down a server.

On network database servers (dbsrv17), the default setting requires the SERVER OPERATOR system privilege to stop a database server. The -gk option allows you to limit access to this option to users with a certain level of privilege in the database.

Loading and Unloading Data

The LOAD TABLE, UNLOAD TABLE, and UNLOAD statements all access the file system on the database server computer. The -gl database server option allows you to control the database privileges required to perform loading and unloading of data.

The default setting is DBA for the network database server. If you are running the network database server, unwarranted file system access may be a security issue.

The default setting is *all* for personal database servers (dbeng17) on non-Unix operating systems. The default setting is DBA on Unix operating systems. If you are running the personal database server, you already have access to the file system and this is not a security issue.

Using Transport Layer Security (TLS) to Encrypt Client/Server Communications

For greater security of network packets, you can use transport layer security to authenticate and encrypt communications between client applications and the database server. Transport layer security uses RSA encryption technology.

Turn on Database Isolation Use the -edi database server option to enable database isolation. When database isolation is enabled, every database running on the database server behaves as though it's the only database running.

Related Information

Security: Passwords [page 1685]
How to Run the Database Server as a Service or Daemon [page 354]
Disk Sandboxing [page 1687]
Super-users [page 1558]
Creating Secured Feature Keys [page 1690]
xp_cmdshell System Procedure
xp_startmail System Procedure
xp_startsmtp System Procedure
xp_sendmail System Procedure
xp_stopmail System Procedure
xp_stopsmtp System Procedure
DROP PROCEDURE Statement
SATMP Environment Variable [page 578]
-ep Database Server Option [page 423]
-ek Database Option [page 551]
-xs Database Server Option [page 529]
request_timeout Option [page 810]
-sbx Database Server Option [page 491]
-sbx Database Option [page 559]
sa_server_option System Procedure
-sf Database Server Option [page 493]

1.9.2.3 Security: User IDs

By assigning user IDs and passwords, the database administrator controls who has access to a database.

By granting privileges to each user ID, the database administrator controls which tasks each user can perform when connected to the database.

Privilege Scheme is Based on User IDs

When a user logs on to the database, they have access to all database objects that meet any of the following criteria:

- objects the user created
- objects to which the user has received explicit privilege
- objects to which a group the user belongs to received explicit privilege

The user cannot access any database object that does not meet these criteria. In short, users can access only the objects they own or objects to which they explicitly received access privileges.

Roles and Privileges

You can control the tasks users can perform on database objects (such as creating, modifying, executing, updating, and so on), and the administrative tasks (such as backing up, profiling, and so on) that a user can perform, by granting roles and privileges.

You grant roles and privileges using the GRANT and GRANT ROLE statement.

The REVOKE and REVOKE ROLE statements perform the opposite of granting. Any role or privilege that GRANT has explicitly given, REVOKE can take away. Revoking CONNECT from a user removes the user from the database, including all objects owned by that user.

Guest User ID

Creating the user ID Guest with a password permits login access to the database for anyone that can authenticate to a database using Integrated or Kerberos logins. No login mapping using a GRANT statement is required. Although the Guest user is limited by the roles and privileges assigned to it, creating the Guest user ID is not recommended.

Related Information

Security: Integrated Logins Can Result in Unrestricted Database Access [page 172]
Microsoft Windows Integrated Login [page 129]

Kerberos User Authentication [page 159]
GRANT Statement
GRANT ROLE Statement
GRANT INTEGRATED LOGIN Statement
GRANT KERBEROS LOGIN Statement
REVOKE Statement
REVOKE ROLE Statement
login_mode Option [page 752]

1.9.2.4 Security: Passwords

Passwords are an important part of any database security system.

To be secure, passwords must be difficult to guess, and they must not be easily accessible on users' hard drives or other locations. SQL Anywhere passwords are case sensitive.

Passwords sent between a client and the database server during a connection are encrypted.

Implement a Login Policy

Use a login policy to control the frequency of password changes and to specify the number of login attempts that are allowed before an account is locked.

Implement Minimum Password Lengths

By default, passwords must be 6 bytes in length. For greater security, change the minimum length requirement on all new passwords to disallow short (and therefore easily guessed) passwords. Do this by setting the min_password_length database option to a value greater than zero. The following statement enforces passwords to be at least 8 bytes long:

```
SET OPTION PUBLIC.min_password_length = 8;
```

Implement Password Expiration

By default, database passwords never expire. Use a login policy to implement password expiry.

Do Not Include Passwords in Procedures

Some statements, such as CREATE USER, have an IDENTIFIED BY clause for specifying a password. When you create a procedure that includes these statements, do not specify the password as a string literal; the

definitions of procedures are visible in the SYSPROCEDURE system view. Instead, create a variable outside of the body of the procedure, and then reference the variable from the IDENTIFIED BY clause.

Do Not Include Passwords in ODBC Data Sources

Passwords should not be easily available to unauthorized people in a security-conscious environment.

When you create profiles that allow users to connect, such as an ODBC data source, do not include passwords. This practice ensures that they are not viewed by unauthorized users.

Encrypt Configuration Files Containing Passwords

When you create a configuration file, do not include passwords. If you decide that you cannot avoid including password information, then consider encoding the contents of the file with the File Hiding utility (dbfhide) and consider securing the file from unauthorized access using appropriate operating system permissions.

Use Password Verification

Use the verify_password_function option to specify a function that implements password rules.

Prevent Any Single Administrator From Knowing the Complete Password of Another User

Use the dual control password feature to require two administrators to change a password. One administrator sets the first part of a password, and another administrator sets the second part of the password. The user specifies the two password parts to connect to the database. The target user must have the change_password_dual_control option enabled in their login policy.

Related Information

Login Policies [page 640]
ODBC Data Sources [page 116]
Setting a Dual Control Password (SQL) [page 1639]
verify_password_function Option [page 866]
CREATE LOGIN POLICY Statement
min_password_length Option [page 772]
File Hiding Utility (dbfhide) [page 1160]

1.9.2.5 Disk Sandboxing

Enabling disk sandbox settings restricts the read-write file operations of the database to the directory where the main database file is located and any subdirectories of this directory.

These restrictions allow users to perform file operations while preventing them from accessing files that users should not have access to. All database users have the same sandbox location.

When disk sandboxing is enabled, relative path names are treated as relative to the directory where the main database file (the system dbspace) is located. When disk sandboxing is not enabled, relative path names are assumed to be relative to the current working directory of the database server.

The most secure setting for disk sandboxing always takes precedence. If a database is started with sandboxing enabled (specified by one of the -sbx server options or the START DATABASE statement or if the disk_sandbox database option is set to On), then the database runs in a disk sandbox regardless of other settings that disable the disk sandboxing feature. Disk sandboxing can be enabled and disabled in the following ways:

Setting	Scope	Persistence	Notes
-sbx database server option	All databases running on the database server	Current session only	• Controlled by the secure feature setting • Specified when starting a database server
DiskSandbox option for the sa_server_option system procedure	All databases running on the database server	Current session only	• Controlled by the secure feature setting • Specified after a database server has started
-sbx database option	An individual database running on the database server	Current session only	• Controlled by the secure feature setting • Specified when starting a database server
sa_db_option system procedure	An individual database running on the database server	Current session only	• Controlled by the secure feature setting • Specified after a database server has started
DISKSANDBOX clause of the START DATABASE statement	An individual database running on the database server	Current session only	• Controlled by the secure feature setting • Specified after a database server has started
disk_sandbox database option	The current database	Setting persists across session	• You must have the required system privilege • Specified for the database

Secure Features That Control Disk Sandboxing

When you start a database server, the manage_disk_sandbox and disk_sandbox secure features are enabled by default. These settings control the ability of users to change disk sandboxing behavior. If you need to disable

the manage_disk_sandbox secure feature for an individual connection, you must use the sp_use_secure_feature_key system procedure.

Database Backups and Disk Sandboxing

If a database is running in a disk sandbox, you can make client-side backups without any restrictions.

To make a server-side backup and store the backup outside the disk sandbox directory, specify the database server secure feature key for the connection that is going to run the backup.

Related Information

-sbx Database Server Option [page 491]
-sbx Database Option [page 559]
disk_sandbox Option [page 730]
START DATABASE Statement
-sf Database Server Option [page 493]
sa_db_option System Procedure
sa_server_option System Procedure
sp_use_secure_feature_key System Procedure

1.9.2.6 Secured Features

Features can be made inaccessible to databases running on a database server. These features are secured from use.

When a feature is secured (made inaccessible), it is unavailable for use by client applications, database-defined stored procedures, triggers, and events. Secured feature settings apply to all databases running on the database server. Secured features are useful when you need to start a database that might contain embedded logic that references the external environment of the host computer system (for example, directories and files on the host computer). This capability is useful to third-party vendors who run a database server farm to host databases from various customers.

The -sf database server option allows you to specify which features you want to secure for databases running on the database server.

Features That Are Secured by Default

When you start a server using the -sf database server option, the following feature sets are secured by default:

- MANAGE_SECURITY

- SERVER_SECURITY (except for TRACE_SYSTEM_EVENT)

Secured Feature Keys

The **SYSTEM secured feature key** is created by specifying the -sk database server option and an authorization code when starting the database server. You specify this authorization code to the sp_use_secure_feature_key system procedure to gain access to the SYSTEM secured feature key. Then you use the sa_server_option system procedure to alter which features are secured or unsecured as the database server is running.

If you start a database server without specifying the -sk option, you cannot change the secured feature settings for the database server or any databases running on it. You cannot create the SYSTEM secured feature key at a later time using the sp_create_secure_feature_key system procedure. Instead, you must shut down the database server and specify the -sk option when you restart it.

Once you have accessed the SYSTEM secured feature key, you can use the sp_create_secure_feature_key system procedure to create **customized secured feature keys** with authorization codes that can be used by other users to gain access to a specific set of features that are otherwise secured from use.

There is a limit of 1000 secured feature keys per database server.

The authorization code must be a non-empty string of at least six characters, and it cannot contain double quotes, control characters (any character less than 0x20), or backslashes.

The SYSTEM and customized secured feature keys are accessed by using the following system procedure:

- sp_use_secure_feature_key system procedure

EXECUTE privilege is required to use this system procedure.

Access is session based. Once you disconnect from the database server, you lose access. When you reconnect to the server, you must reacquire access.

Features are secured and unsecured by using the following system procedure:

- sa_server_option system procedure with the 'SecureFeatures' option.

Features are secured or unsecured while the database server is running. If the server is shut down, the settings are lost.

Customized secured feature keys are managed by using the following system procedures:

- sp_create_secure_feature_key system procedure
- sp_alter_secure_feature_key system procedure
- sp_drop_secure_feature_key system procedure
- sp_list_secure_feature_keys system procedure

The MANAGE_KEYS feature must be enabled (acquired) to use these system procedures.

Secured feature keys exist while the database server is running. If the server is shut down, the keys are lost. The keys must be recreated when the server is restated.

> **i** Note
>
> The SYSTEM secured feature key can only be dropped if a customized secured feature key has been created that has both the MANAGE_FEATURES and MANAGE_KEYS features enabled.

In this section:

Related Information

-sf Database Server Option [page 493]
-sk Database Server Option [page 500]
sa_server_option System Procedure
sp_alter_secure_feature_key System Procedure
sp_create_secure_feature_key System Procedure
sp_drop_secure_feature_key System Procedure
sp_list_secure_feature_keys System Procedure
sp_use_secure_feature_key System Procedure
sp_copy_directory System Procedure
sp_copy_file System Procedure
sp_create_directory System Procedure
sp_delete_directory System Procedure
sp_delete_file System Procedure
sp_list_directory System Procedure
sp_move_directory System Procedure
sp_move_file System Procedure

1.9.2.6.1 Creating Secured Feature Keys

Control the database features available to users, by using the secure features database server option (-sf) to specify the features that users are prevented from accessing on the database server.

Prerequisites

You must have the SERVER OPERATOR system privilege and have access to the MANAGE_KEYS feature.

Context

Secured feature settings apply to all databases running on a database server.

The secure features option (-sf) controls the availability of such features as:

- Server-side backups
- External stored procedures
- Remote data access
- Web services

The -sk option specifies a SYSTEM secured feature key that manages access to secured features for a database server. To alter the list of secured features once the database server is running, use the sa_server_option system procedure. To alter a customized secured feature key once the database server is running, use the sp_alter_secure_feature_key system procedure.

The sp_create_secure_feature_key system procedure creates a customized secured feature key.

Procedure

1. At a command prompt, start the database server using the -sf and -sk options.

   For example, the following command starts the database server and secures all features. The command also includes a key that can be used later to allow access to secured features for a connection.

   ```
   dbsrv17 -n secure_server -sf all -sk secretAuthCode mydemo.db
   ```

2. Connect to the database server:

   ```
   dbisql -c "UID=DBA;PWD=passwd;Host=myhost;Server=secure_server;DBN=mydemo"
   ```

3. Call the sp_use_secure_feature_key system procedure to specify the SYSTEM secured feature key for the connection. The authorization code to use is specified by the -sk option:

   ```
   CALL sp_use_secure_feature_key ( 'system' , 'secretAuthCode' );
   ```

4. Change the set of secured features on the server by using the sa_server_option system procedure.

   For example:

   ```
   CALL sa_server_option( 'SecureFeatures', 'all,-remote_data_access' );
   ```

5. Create a customized secured feature key for a specific user.

   For example, create a customized secured feature key for Bob that allows him to send emails:

   ```
   CALL sp_create_secure_feature_key ( 'bobsKey' , 'anotherAuthKey' ,
   'sa_send_email' );
   ```

   After logging into the database, Bob must run the following command to send emails:

   ```
   CALL sp_use_secure_feature_key ( 'bobsKey' , 'anotherAuthKey' );
   ```

Results

There is now a SYSTEM secured feature key for the database server, as well as a customized secured feature key that has been assigned to a specific user.

Users of databases running on the database server secure_server are prevented from accessing all secured features except the remote_data_access feature. The user Bob, however, also has access to the sa_send_email feature.

Related Information

-sf Database Server Option [page 493]
-sk Database Server Option [page 500]
sa_server_option System Procedure
sp_alter_secure_feature_key System Procedure
sp_create_secure_feature_key System Procedure
sp_drop_secure_feature_key System Procedure
sp_list_secure_feature_keys System Procedure
sp_use_secure_feature_key System Procedure
-sf Database Server Option [page 493]
-sk Database Server Option [page 500]

1.9.2.7 Database Activity Audits

Auditing tracks all of the activity performed on a database.

Each database has an associated transaction log file that is used for database recovery and contains a record of transactions executed against a database.

The transaction log stores all executed data definition statements, along with the user ID that executed them. It also stores all updates, deletes, and inserts and which user executed those statements. However, this information is insufficient for some auditing purposes. By default, the transaction log does not contain the time of the event. It only contains the order in which events occurred. It does not contain failed events or select statements.

When you use auditing, additional data is saved in the audit log, which is in the transaction log, the user-defined ETD log, or the system event log. This data potentially includes:

- All login attempts (successful and failed), including the computer name.
- Accurate timestamps of all events (to a resolution of milliseconds).
- All permissions checks (successful and failed), including the object on which the permission was checked (if applicable).
- All actions that require system privileges.
- All executions of xp_cmdshell system procedure.

You cannot stop using a transaction log while auditing is enabled for a database and the audit log is the transaction log. In this case, to turn off the transaction log, either first turn off auditing or change the audit_log option to specify FILE or SYSLOG.

Auditing Individual Connections

Once you have enabled auditing for a database, you can set the temporary conn_auditing database option in the database login procedure to enable connection-specific auditing. You can enable auditing based on information such as the IP address of the client computer or the type of connection.

If you do not set the conn_auditing option in the login procedure, then the option is on by default.

The following example shows an excerpt from a login procedure that enables auditing for all connections to the database, except those made by a user named DBA:

```
DECLARE usr VARCHAR(128)
SELECT CONNECTION_PROPERTY( 'Userid' ) INTO usr;
IF usr != 'DBA' THEN
   SET TEMPORARY OPTION conn_auditing='On'
ELSE
   SET TEMPORARY OPTION conn_auditing='Off'
END IF;
```

Audit Comments

Add comments to the audit trail by using the sa_audit_string system stored procedure. It takes a single argument – a VARCHAR(128) string. For example:

```
CALL sa_audit_string( 'Started audit testing here.' );
```

This comment is stored in the audit log as an audit statement.

In this section:

Configuring Auditing (SQL) [page 1694]
Configure auditing for security-related information in the transaction log. Auditing is off by default.

Retrieving Auditing Information (dbtran Utility) [page 1695]
Retrieve auditing information from a transaction log.

Tutorial: Auditing [page 1696]
Attempt to access unauthorized information and then view how the attempt is recorded by the auditing feature.

Related Information

The Transaction Log [page 292]

sa_audit_string System Procedure

1.9.2.7.1 Configuring Auditing (SQL)

Configure auditing for security-related information in the transaction log. Auditing is off by default.

Prerequisites

You must have the SET ANY SECURITY OPTION system privilege.

Procedure

1. Connect to your database.
2. Execute the following statement to turn on auditing:

   ```
   SET OPTION PUBLIC.auditing = 'On';
   ```
3. Execute the following statement to force logging to a transaction log:

   ```
   SET OPTION PUBLIC.audit_log = 'TRANSLOG';
   ```
4. To specify which types of auditing information you want to enable, run the following system procedure:

   ```
   CALL sa_enable_auditing_type( 'all' );
   ```

   You can control the type of auditing information that is collected by replacing *all* with the types of auditing you want to enable.
5. Execute the following statement to turn off auditing:

   ```
   SET OPTION PUBLIC.auditing = 'Off';
   ```

   To specify which types of auditing information you want to disable, use the following system procedure:

   ```
   CALL sa_disable_auditing_type( 'all' );
   ```

   You can stop collecting specific types of auditing information by replacing *all* with the types of auditing you want to disable.

Results

The auditing behavior is configured.

Related Information

login_procedure Option [page 755]
conn_auditing Option [page 706]
auditing Option [page 688]
auditing Option [page 688]
sa_enable_auditing_type System Procedure
auditing_options Option (Reserved for System Use) [page 690]

1.9.2.7.2 Retrieving Auditing Information (dbtran Utility)

Retrieve auditing information from a transaction log.

Prerequisites

You must have the BACKUP DATABASE system privilege to retrieve auditing information from a running database server. No privileges are required to retrieve auditing information from a database transaction log file.

The database must have the auditing database option turned on.

Context

The dbtran utility translates a transaction log into a SQL script file. When the dbtran utility is run with the -g option, it adds auditing information (as comments) and orders the SQL statements chronologically. Do not apply this output to a database, because running statements in chronological order does not guarantee that the resulting database will be identical.

Procedure

1. Connect to the database.
2. Run the dbtran utility with the -g option against a running database server or against a database transaction log file.

Option	Action
Retrieve auditing information from a running database server	Run the following command: `dbtran -g -c connection-string -n SQL-file`

Option	Action
Retrieve auditing information from a transaction log file	Shut down the database server to ensure the transaction log file is available. Run the following command: `dbtran -g transaction-log SQL-file`

Results

Auditing information is retrieved and saved in a SQL script file.

Example

In this example, the auditing information is saved to the *demo*`.sql` file, and the file contains information about the sample database.

```
dbtran -g -c "UID=DBA;PWD=sql;DBN=demo" -n demo.sql
```

In the following example, the auditing information from the transaction log file *demo*`.log` is placed into the file *demo*`.sql`.

```
dbtran -g demo.log demo.sql
```

Related Information

Alphabetical List of Connection Parameters [page 45]
Log Translation Utility (dbtran) [page 1187]

1.9.2.7.3 Tutorial: Auditing

Attempt to access unauthorized information and then view how the attempt is recorded by the auditing feature.

Prerequisites

You must have the SET ANY SECURITY OPTION and MANAGE ANY USER system privileges

Procedure

1. Start Interactive SQL and connect to the sample database using the SQL Anywhere 17 Demo data source.
2. Turn on auditing using the SET OPTION statement, as follows:

   ```
   SET OPTION PUBLIC.auditing = 'On';
   ```

3. Add a user, Test1, to the sample database using the CREATE USER statement, as follows:

   ```
   CREATE USER Test1
   IDENTIFIED BY welcome;
   ```

4. Open a new Interactive SQL window, connect to the sample database as Test1, and attempt to access confidential information in the Employees table using the following SELECT statement:

   ```
   SELECT Surname, Salary
      FROM GROUPO.Employees;
   ```

   An error message appears indicating that you do not have permission to select from the Employees table.
5. Run the following command to view the auditing information in `demo.sql` for this activity:

   ```
   dbtran -g -c "DSN=SQL Anywhere 17 Demo;PWD=sql" -n demo.sql
   ```

6. Restore the sample database to its original state:
   a. Use the DROP USER statement to remove the Test1 user from the database:

      ```
      DROP USER Test1;
      ```

   b. Turn off auditing using the following SET OPTION statement:

      ```
      SET OPTION PUBLIC.auditing = 'Off';
      ```

1.9.2.8 Database Encryption and Decryption

Make it more difficult for someone to decipher the data in your database.

You can choose to secure your database either with simple obfuscation or with strong encryption.

The database administrator has control over four aspects of strong encryption, including:

- What is encrypted (database, dbspaces, individual tables, transaction log, transaction log mirror)
- The encryption key
- The protection of the encryption key
- Encryption algorithm

If your database is obfuscated or encrypted, compressing it with a tool such as WinZip does not result in a file that is significantly smaller than the original database file.

In this section:

How to Encrypt a Database [page 1699]
There are several ways to encrypt a database.

How to Encrypt Tables, Columns, and Materialized Views [page 1702]
If you only want to encrypt portions of your database, you can choose to encrypt columns or tables.

Security: Keys For Encrypted Databases [page 1712]
Learn how to increase the security of the encryption keys used with strongly encrypted databases and tables.

Creating an Encrypted Database (SQL) [page 1715]
Encrypt a database during creation.

Creating an Encrypted Database (dbinit Utility) [page 1716]
Encrypt the database when you create the database.

Creating a Database That Is Encoded Using Simple Obfuscation [page 1717]
Specify the encoding option when creating the database.

Encrypting an Existing Database (CREATE ENCRYPTED DATABASE Statement) [page 1718]
Create a strongly encrypted copy of a database along with encrypted copies of any associated transaction logs, transaction log mirrors, or dbspace files.

Encrypting an Existing Database (CREATE ENCRYPTED FILE Statement) [page 1720]
Create an encrypted copy of an existing database and its associated files when you cannot use the CREATE ENCRYPTED DATABASE statement. For example, you need to encrypt your database for support purposes and the CREATE ENCRYPTED DATABASE statement returns an error.

Decrypting a Database (SQL) [page 1721]
Decrypt a database along with any associated transaction log, transaction log mirror, and dbspace files.

Performance Issues When Using Encryption [page 1723]
Performance is slower when the database is encrypted. The performance impact depends on how often pages are read from or written to disk, and can be minimized by ensuring that the server is using an adequate cache size.

1.9.2.8.1 Simple Obfuscation Versus Strong Encryption

Two methods of database encoding are supported: simple obfuscation and strong encryption.

Simple Obfuscation

Simple obfuscation is intended to make it difficult, but not impossible, for someone using a disk utility to casually inspect the contents of your database. Simple obfuscation does not require a key (password) to encode the database.

Strong Encryption (AES and ARIA)

Strong encryption makes a database unusable without a key (password). The data in the database is secure from inspection. An algorithm encrypts the information contained in your database and transaction log files so it cannot be read.

The algorithms used to implement strong encryption are AES and ARIA.

You can indicate the use of a 128-bit encryption algorithm using the AES keyword. You can indicate the use of a 256-bit encryption algorithm using the AES256, AES256CTR, or ARIA256 keywords. This encryption technology is included and does not require a separate license.

You can also indicate the use of a separately licensed FIPS-certified AES module for strong encryption by specifying one of the AES_FIPS (128-bit algorithm) or AES256_FIPS (256-bit algorithm) keywords. When the database server is started with the -fips option, you can start databases encrypted with any of the AES, AES256, AES_FIPS, AES256_FIPS, or AES256CTR_FIPS strong encryption algorithms, but not databases encrypted with simple obfuscation. Unencrypted databases can also be started on the server when -fips is specified.

To start a database encrypted with AES_FIPS, AES256_FIPS or AES256CTR_FIPS, the separately licensed FIPS-certified AES module must be installed on the computer.

All strong encryption technologies are subject to export regulations.

Related Information

Separately Licensed Components
Reloading a Database
-fips Database Server Option [page 426]
Initialization Utility (dbinit) [page 1166]
CREATE DATABASE Statement
CREATE ENCRYPTED DATABASE Statement

1.9.2.8.2 How to Encrypt a Database

There are several ways to encrypt a database.

To create an encrypted database

You can use the following:

- The Initialization utility (dbinit), with options such as -ep, -ek, or -ea to enable strong encryption.
- CREATE DATABASE statement.
- The SQL Central *Create Database Wizard* to create a strongly encrypted database.

To encrypt an existing database

Although you cannot simply turn strong encryption on or off in an existing database, you can use one of the following to create a strong-encrypted copy of your database:

- You can use the CREATE ENCRYPTED DATABASE statement. The CREATE ENCRYPTED DATABASE statement is the preferred method because it encrypts the database as well as associated files, such as the transaction log, transaction log mirror, and dbspace files.
- You can use the CREATE ENCRYPTED FILE statement when it is not possible to use the CREATE ENCRYPTED DATABASE statement. After you have encrypted the database with this statement, you must encrypt any associated files, such as the transaction log, transaction log mirror, and dbspace files.

In this section:

Comparison of the CREATE ENCRYPTED DATABASE and CREATE ENCRYPTED FILE Statements [page 1701]
Use the CREATE ENCRYPTED DATABASE statement when you have an existing database that you want to encrypt. Use the CREATE ENCRYPTED FILE statement when you have a database you want to encrypt for support purposes and it is not possible to use the CREATE ENCRYPTED DATABASE statement.

Encryption Algorithm Aliases [page 1701]
The encryption algorithms for SAP SQL Anywhere database encryption have multiple aliases for compatibility across systems.

Related Information

How to Encrypt Tables, Columns, and Materialized Views [page 1702]
Tips on Exporting Data with the Unload Database Wizard
Comparison of the CREATE ENCRYPTED DATABASE and CREATE ENCRYPTED FILE Statements [page 1701]
Message Encryption Using Public and Private Keys [page 1723]
Initialization Utility (dbinit) [page 1166]
CREATE DATABASE Statement
ENCRYPT Function [String]
Unload Utility (dbunload) [page 1233]
UNLOAD Statement
CREATE ENCRYPTED DATABASE Statement
CREATE ENCRYPTED FILE Statement

1.9.2.8.2.1 Comparison of the CREATE ENCRYPTED DATABASE and CREATE ENCRYPTED FILE Statements

Use the CREATE ENCRYPTED DATABASE statement when you have an existing database that you want to encrypt. Use the CREATE ENCRYPTED FILE statement when you have a database you want to encrypt for support purposes and it is not possible to use the CREATE ENCRYPTED DATABASE statement.

You cannot be connected to the database you are encrypting when you execute the statement.

The CREATE ENCRYPTED FILE and CREATE ENCRYPTED DATABASE statements differ from each other as follows:

- The CREATE ENCRYPTED FILE statement must be executed against each of the database-related files independently (transaction log, transaction log mirror, dbspaces, if any), whereas the CREATE ENCRYPTED DATABASE statement automatically encrypts all the database-related files.
- The CREATE ENCRYPTED DATABASE statement cannot be used on a database requiring recovery; the CREATE ENCRYPTED FILE statement can.
- The CREATE ENCRYPTED DATABASE statement cannot be used inside procedures, triggers, or batches. The CREATE ENCRYPTED FILE statement can.
- The CREATE ENCRYPTED DATABASE statement supports the SIMPLE obfuscation algorithm, but the CREATE ENCRYPTED FILE statement does not.

Related Information

Supported Platforms
Encrypting an Existing Database (CREATE ENCRYPTED DATABASE Statement) [page 1718]
CREATE ENCRYPTED DATABASE Statement
CREATE ENCRYPTED FILE Statement
DatabaseKey (DBKEY) Connection Parameter [page 70]

1.9.2.8.2.2 Encryption Algorithm Aliases

The encryption algorithms for SAP SQL Anywhere database encryption have multiple aliases for compatibility across systems.

Encryption Algorithm	Supported Aliases	SAP HANA Database-Compatible Alias	Description
AES	AES128; AES_128; AES-128; AES-128-CBC	AES-128-CBC	AES 128-bit algorithm with CBC block cipher mode
AES_FIPS	AES128_FIPS; AES-128_FIPS;		FIPS-certified version of the AES 128-bit algorithm with CBC block cipher mode

Encryption Algorithm	Supported Aliases	SAP HANA Database-Compatible Alias	Description
	AES-128_FIPS; AES-128-CBC_FIPS		
AES256	AES-256-CBC; AES-256; AES_256; AES_256_CBC	AES-256-CBC	AES 256-bit algorithm with CBC block cipher mode
AES256_FIPS	AES-256-CBC_FIPS; AES-256_FIPS; AES_256_FIPS; AES_256_CBC_FIPS		FIPS-certified version of the AES 256-bit algorithm with CBC block cipher mode
AES256CTR	AES-256-CTR; AES_256_CTR; AES256_CTR; AES256-CTR	AES-256-CTR	AES 256-bit algorithm with CTR block cipher mode
AES256CTR_FIPS	AES-256-CTR_FIPS; AES_256_CTR_FIPS; AES256_CTR_FIPS; AES256-CTR_FIPS		FIPS-certified version of the AES 256-bit algorithm with CTR block cipher mode
ARIA256	ARIA_256_CBC; ARIA-256; ARIA_256; ARIA256CBC; ARIA256_CBC	ARIA-256-CBC	ARIA 256-bit algorithm with CBC block cipher mode
ARIA256CTR	ARIA-256-CTR; ARIA_256_CTR; ARIA256_CTR; ARIA256-CTR	ARIA-256-CTR	ARIA 256-bit algorithm with CTR block cipher mode

1.9.2.8.3 How to Encrypt Tables, Columns, and Materialized Views

If you only want to encrypt portions of your database, you can choose to encrypt columns or tables.

Column encryption can be performed on any column in any table at any time. Table encryption requires that the database have table encryption enabled. Table encryption is enabled at database creation (initialization) time.

To encrypt tables

You can use the following:

- Initialization utility (dbinit).
- CREATE DATABASE statement.
- CREATE ENCRYPTED TABLE DATABASE statement.

To encrypt columns

ENCRYPT function.

To encrypt materialized views

ALTER MATERIALIZED VIEW statement.

In this section:

Related Information

Encrypting or Decrypting a Materialized View
Initialization Utility (dbinit) [page 1166]
CREATE DATABASE Statement
ALTER DATABASE Statement
CREATE ENCRYPTED DATABASE Statement
ENCRYPT Function [String]
ALTER MATERIALIZED VIEW Statement

1.9.2.8.3.1 Column Encryption

To encrypt columns in your database, use the ENCRYPT function.

The ENCRYPT function uses the same AES strong encryption algorithm that is used for database encryption to encrypt values that are passed to it.

Encrypted data can be decrypted with the DECRYPT function. You must use the same key that was specified in the ENCRYPT function. Both of these functions return LONG BINARY values. If you require a different data type, you can use the CAST function to convert the value to the required data type.

The ENCRYPT and DECRYPT functions also support raw encryption. You can encrypt data inside the database server into a format that can be exported and decrypted outside of the server.

If database users need to access the data in decrypted form, but you do not want them to have access to the encryption key, you can create a view that uses the DECRYPT function. This allows users to access the decrypted data without knowing the encryption key. If you create a view or stored procedure that uses the table, you can use the SET HIDDEN parameter of the ALTER VIEW and ALTER PROCEDURE statements to ensure that users cannot access the encryption key by looking at the view or procedure definition.

Column Encryption Example

The following example uses triggers to encrypt a column that stores passwords in a table called user_info. The user_info table is defined as follows:

```
CREATE TABLE user_info (
   employee_ID INTEGER NOT NULL PRIMARY KEY,
   user_name CHAR(80),
   user_pwd CHAR(80) );
```

Two triggers are added to the database to encrypt the value in the user_pwd column, either when a new user is added or an existing user's password is updated.

- The encrypt_new_user_pwd trigger fires each time a new row is added to the user_info_table:

  ```
  CREATE TRIGGER encrypt_new_user_pwd
  BEFORE INSERT
  ON user_info
  REFERENCING NEW AS new_pwd
  FOR EACH ROW
  BEGIN
      SET new_pwd.user_pwd=ENCRYPT(new_pwd.user_pwd, '8U3dkA');
  END;
  ```

- The encrypt_updated_pwd trigger fires each time the user_pwd column is updated in the user_info table:

  ```
  CREATE TRIGGER encrypt_updated_pwd
  BEFORE UPDATE OF user_pwd
  ON user_info
  REFERENCING NEW AS new_pwd
  FOR EACH ROW
  BEGIN
      SET new_pwd.user_pwd=ENCRYPT(new_pwd.user_pwd, '8U3dkA');
  END;
  ```

Add a new user to the database:

```
INSERT INTO user_info
VALUES ( '1', 'd_williamson', 'abc123');
```

If you issue a SELECT statement to view the information in the user_info table, the value in the user_pwd column is binary data (the encrypted form of the password) and not the value abc123 that was specified in the INSERT statement.

If this user's password is changed, then the encrypt_updated_pwd trigger fires and the encrypted form of the new password appears in the user_pwd column.

```
UPDATE user_info
SET user_pwd='xyz'
WHERE employee_ID='1';
```

The original password can be retrieved by issuing the following SQL statement. This statement uses the DECRYPT function and the encryption key to decrypt the data, and the CAST function to convert the value from a LONG BINARY to a CHAR value:

```
SELECT CAST (
   DECRYPT( user_pwd, '8U3dkA' )
   AS CHAR(100))
   FROM user_info
```

```
    WHERE employee_ID = '1';
```

In this section:

Related Information

CAST Function [Data Type Conversion]
ALTER PROCEDURE Statement
ALTER VIEW Statement
ENCRYPT Function [String]
DECRYPT Function [String]

1.9.2.8.3.1.1 Raw Encryption

Raw encryption allows you to encrypt data inside the database server into a format that can be exported and decrypted outside of the database server.

The encrypted format is referred to as **raw**. To encrypt data in the raw format, you must specify the encryption key, the initialization vector, and optionally a padding format. To decrypt the data, you must specify the same parameter values.

You can also use the DECRYPT function to decrypt the data inside the database server.

Raw encryption is useful when:

You want to prevent database users from having access to the data

You can use raw encryption to encrypt sensitive data that you do not want even your database administrators to have access to, and then decrypt the data using a client application without the use of the database server. Raw encryption is not recommended when the data needs to be encrypted and decrypted only by the database server.

You cannot use TLS encryption

You can use raw encryption instead of TLS encryption. Unlike TLS encryption, raw encryption cannot prevent replay or person-in-the-middle attacks, nor can it authenticate database servers.

Example

You need to send data from the binary_data column of the SensitiveData table in your database to a client that does not use databases. Because the data is sensitive, you encrypt the data into raw format using the following SQL statement:

```
SELECT ENCRYPT( binary_data, 'TheEncryptionKey','AES(FORMAT=RAW)',
'ThisIsTheIV' )FROM SensitiveData;
```

You copy the encrypted data to the client along with an application that can decrypt the contents. You also provide the encryption key (TheEncryptionKey) and the initialization vector (ThisIsTheIV) to the client to use with the application. The client uses the application to decrypt the data and view it.

Related Information

ENCRYPT Function [String]
DECRYPT Function [String]

1.9.2.8.3.2 Table Encryption

Table encryption allows you to encrypt tables or materialized views with sensitive data, without the performance impact that encrypting the entire database might cause.

When table encryption is enabled, table pages for the encrypted table, associated index pages, and temporary file pages are encrypted. The transaction log pages that contain transactions on encrypted tables are also encrypted.

To encrypt tables in your database, you must have table encryption enabled. Enabling table encryption must be done at database initialization. To see whether table encryption is enabled, query the EncryptionScope database property using the DB_PROPERTY function, as follows:

```
SELECT DB_PROPERTY( 'EncryptionScope' );
```

If the return value is TABLE, table encryption is enabled.

To see the encryption algorithm in effect for table encryption, query the Encryption database property using the DB_PROPERTY function, as follows:

```
SELECT DB_PROPERTY( 'Encryption' );
```

Performance Impact of Table Encryption

For encrypted tables, each table page is encrypted when written to the disk, and is decrypted when read in from the disk. This process is invisible to applications. However, there may be a slight negative impact on

performance when reading from, or writing to, encrypted tables. Encrypting or decrypting existing tables can take a long time, depending on the size of the table.

Index pages for indexes on columns in an encrypted table are also encrypted, as are transaction log pages containing transactions on the encrypted table, and all pages in the temporary file for the database. All other database and transaction log pages are unencrypted.

Encrypted tables can contain compressed columns. In this case, the data is compressed before it is encrypted.

Encrypting tables does not impact storage requirements.

Starting a Database That Has Table Encryption Enabled

Starting a database that has table encryption enabled is the same as starting an encrypted database. For example, if the database is started with the -ek option, a key must be specified. If the database is started with the -ep option, you are prompted for the key.

Related Information

Database Encryption and Decryption [page 1697]
How to Encrypt a Database [page 1699]
Encrypting or Decrypting a Materialized View
Creating a Database (SQL Central) [page 278]
Creating a Database (dbinit Utility) [page 279]
Initialization Utility (dbinit) [page 1166]
-ep Database Server Option [page 423]
ALTER TABLE Statement
CREATE DATABASE Statement
CREATE ENCRYPTED DATABASE Statement
CREATE DECRYPTED DATABASE Statement
CREATE TABLE Statement

1.9.2.8.3.3 Enabling Table Encryption in a Database (SQL)

Create a database with table encryption enabled, or enable table encryption in an existing database, so that you can later encrypt your tables.

Prerequisites

By default, you must have the SERVER OPERATOR system privilege to execute the CREATE DATABASE statement and the CREATE ENCRYPTED TABLE DATABASE statement. The required privileges can be changed by using the -gu database server option.

Context

Table encryption must be enabled and configured at database creation time. If your database does not have table encryption enabled, or if you have database encryption in effect, using the CREATE ENCRYPTED TABLE DATABASE statement creates a copy of the database with table encryption enabled, and does not overwrite the original database file.

Once table encryption is enabled in the database, you can create encrypted tables or encrypt existing tables.

Procedure

Create a database with table encryption, or enable table encryption on an existing database.

Option	Action
Create a database with table encryption	Create a database with the CREATE DATABASE statement, specify the ENCRYPTED TABLE clause, and specify a key and an encryption algorithm.
Enable table encryption for an existing database	Create a copy of the database with the CREATE ENCRYPTED TABLE DATABASE statement, and specify a key.

Results

Table encryption is enabled.

Example

The following command creates the database `mynewdemo.db` with strong encryption enabled for tables using the key abc, and the AES256_FIPS encryption algorithm. It also creates a DBA user ID with password passwd.

```
CREATE DATABASE 'C:\temp\mynewdemo.db'
   DBA USER 'DBA' DBA PASSWORD 'passwd'
   ENCRYPTED TABLE
   KEY 'abc'
   ALGORITHM 'AES256_FIPS';
```

The following example creates a database called contacts2 from an existing database called contacts1. The new database supports encrypted tables.

```
CREATE ENCRYPTED TABLE DATABASE 'C:\temp\contacts2.db'
   FROM 'C:\temp\contacts1.db'
   KEY 'Sd8f6654'
   OLD KEY 'Sc8e5543';
```

Next Steps

Encrypt tables in the database by using the CREATE TABLE statement, or by altering an existing table to be encrypted by using the ALTER TABLE statement. When you encrypt a table, the key and/or algorithm specified when enabling table encryption is used.

Related Information

Encrypting a Table [page 1711]
CREATE DATABASE Statement
CREATE ENCRYPTED DATABASE Statement
CREATE TABLE Statement
ALTER TABLE Statement

1.9.2.8.3.4 Enabling Table Encryption in a Database (dbinit Utility)

Enable table encryption during the creation of a database.

Prerequisites

Table encryption must be enabled and configured at database creation time. You must re-create the database with table encryption enabled if your database does not have table encryption enabled, or if you have database encryption in effect.

Procedure

Create a database with the dbinit -et and -ek options, and specify a key and an encryption algorithm.

Results

Table encryption is enabled.

Example

The following command creates the database `new.db` with strong encryption enabled for tables using the key abc and the AES256_FIPS encryption algorithm, and sets the DBA user ID to DBA with password passwd:

```
dbinit -dba DBA,passwd new.db -et -ek abc -ea AES256_FIPS
```

Later, when you encrypt a table in this database, the AES256_FIPS algorithm and abc key are used.

Related Information

Encrypting a Table [page 1711]
Initialization Utility (dbinit) [page 1166]
CREATE DATABASE Statement
CREATE ENCRYPTED DATABASE Statement
CREATE TABLE Statement
ALTER TABLE Statement

1.9.2.8.3.5 Encrypting a Table

Create an encrypted table, or encrypt an existing table.

Prerequisites

To use the CREATE TABLE statement, you must have one of the following system privileges:

- CREATE TABLE
- CREATE ANY TABLE
- CREATE ANY OBJECT

To use the ALTER TABLE statement, you must be the owner of the table being altered or have one of the following privileges:

- ALTER privilege on the table
- ALTER ANY TABLE
- ALTER ANY OBJECT

To encrypt tables in your database, table encryption must already be enabled in the database.

Context

When you encrypt a table, the encryption algorithm and key that were specified at database creation time are used.

Procedure

You can either create a table with encryption, or encrypt an existing table.

Option	Action
Create a table with encryption	Create a table using the ENCRYPTED clause of the CREATE TABLE statement.
Encrypt an existing table	Encrypt a table with the ENCRYPTED clause of the ALTER TABLE statement.

Results

The table is encrypted.

Example

The following command creates an encrypted table named MyEmployees:

```
CREATE TABLE MyEmployees (
 MemberID CHAR(40),
 CardNumber INTEGER )
ENCRYPTED;
```

The following statements create a table called MyEmployees2 and then encrypt it.

```
CREATE TABLE MyEmployees2 (
   MemberID CHAR(40),
   CardNumber INTEGER );
ALTER TABLE MyEmployees2
   ENCRYPTED;
```

Related Information

Enabling Table Encryption in a Database (SQL) [page 1708]
Enabling Table Encryption in a Database (dbinit Utility) [page 1710]
CREATE TABLE Statement
ALTER TABLE Statement

1.9.2.8.4 Security: Keys For Encrypted Databases

Learn how to increase the security of the encryption keys used with strongly encrypted databases and tables.

When you encrypt a database or a database table using strong encryption, you must specify an encryption key (password).

Specifying an Encryption Key

The encryption key must be supplied with the -ek option each time the database is started. For example, clients are required to specify the encryption key each time they start the database. If you are concerned about providing your clients with the encryption key, then always have the database administrator start the database.

You can choose whether the encryption key is entered at a command prompt (the default) or into a prompt box. Choosing to enter the key in a prompt box provides an extra measure of security because the key is never visible in plain sight. When starting the database, specify the -ep database option, to be prompted for the encryption key.

Choosing an Encryption Key

When choosing an encryption key, it is best to choose a value that cannot be easily guessed. As well, including a combination of numbers, letters, and special characters decreases the chances of someone guessing the key. Encryption keys are always case sensitive, and they cannot contain leading or trailing spaces or semicolons. Encryption keys can be of arbitrary length. Longer keys are better because a shorter key is easier to guess

Storing Your Encryption Keys

Lost or forgotten keys result in completely inaccessible databases.

> ⚠ Caution
>
> Store a copy of your encryption key in a safe location. If you lose the encryption key, there is no way to access the data, even with the assistance of Technical Support. The database must be discarded and you must create a new database.

In this section:

1.9.2.8.4.1 Changing the Encryption Key For a Database (SQL)

Change the encryption key for an encrypted database, or for a database for which table encryption has been enabled.

Prerequisites

By default, you must have the SERVER OPERATOR system privilege to execute the CREATE ENCRYPTED DATABASE statement. The required privileges can be changed by using the -gu database server option.

Context

Changing the encryption key does not overwrite the existing file, but creates a copy of the file encrypted with the new key. When you change the encryption key you can specify a different algorithm and specify a different number of KDI iterations.

Procedure

1. Change the encryption key for an encrypted database using the CREATE ENCRYPTED DATABASE statement.
2. Choose one of the following options

Option	Action
Change the encryption key for table encryption	Create a copy of the database with the CREATE ENCRYPTED TABLE DATABASE statement, and specify a new key.
Change the encryption key for an encrypted database.	Create a copy of the database with the CREATE ENCRYPTED DATABASE statement, and specify a new key.

Results

The encryption key is changed.

Example

The following example takes the database file `encryptedtemp.db`, encrypted with key abc, and creates a copy of it called `mynewencryptedtemp.db`, encrypting it with the key abc123. Any other database-related files (the transaction log, transaction log mirrors, and dbspace files) are also created using the new encryption key.

```
CREATE ENCRYPTED DATABASE 'C:\temp\mynewencryptedtemp.db'
FROM 'C:\temp\encryptedtemp.db'
KEY 'abc123'
ALGORITHM 'AES'
OLD KEY 'abc';
```

Related Information

CREATE ENCRYPTED DATABASE Statement

1.9.2.8.5 Creating an Encrypted Database (SQL)

Encrypt a database during creation.

Prerequisites

By default, you must have the SERVER OPERATOR system privilege. The required privileges can be changed by using the -gu database server option.

Context

This task is different from encrypting an existing database. To encrypt an existing database, use the CREATE ENCRYPTED DATABASE statement.

> ⚠ Caution
>
> For strongly encrypted databases, store a copy of the key in a safe location. If you lose the encryption key, there is no way to access the data, even with the assistance of Technical Support. The database must be discarded and you must create a new database.

Procedure

1. In Interactive SQL, connect to an existing database.
2. Execute a CREATE DATABASE statement that includes the ENCRYPTED clause and the KEY and ALGORITHM options.

Results

An encrypted database is created.

Example

For example, the following statement creates a database file named `myencrypteddb.db` in the `c:\temp\` directory using FIPS-certified 128-bit AES encryption and the specified encryption key. The DBA user ID and password are specified and a transaction log is enabled.

```
CREATE DATABASE 'c:\\temp\\myencrypteddb.db'
```

```
DBA USER 'DBA'
DBA PASSWORD 'passwd'
TRANSACTION LOG ON
ENCRYPTED ON
  KEY '0kZ2o52AK#'
  ALGORITHM 'AES_FIPS';
```

Related Information

CREATE DATABASE Statement
-gu Database Server Option [page 452]

1.9.2.8.6 Creating an Encrypted Database (dbinit Utility)

Encrypt the database when you create the database.

Context

> ⚠ Caution
>
> For strongly encrypted databases, store a copy of the key in a safe location. If you lose the encryption key, there is no way to access the data, even with the assistance of Technical Support. The database must be discarded and you must create a new database.

Procedure

Run the dbinit utility to create a database.

- To encode the database with strong encryption, include -ek or -ep options to specify the encryption key.

Results

An encoded database is created.

Example

- The following command creates a strongly-encrypted database and specifies the encryption key and algorithm.

  ```
  dbinit -dba DBA,passwd -ek "0kZ2o56AK#" -ea AES_FIPS myencrypteddb.db
  ```

  To start this database, run the following command:

  ```
  dbsrv17 myencrypteddb.db -ek "0kZ2o56AK#"
  ```

Next Steps

When starting a strongly-encrypted database, you must specify the encryption key.

Related Information

Initialization Utility (dbinit) [page 1166]
CREATE DATABASE Statement

1.9.2.8.7 Creating a Database That Is Encoded Using Simple Obfuscation

Specify the encoding option when creating the database.

Context

Procedure

Run the dbinit utility to create a database.

- To encode the database with simple obfuscation, include the -ea simple option.

Results

An encoded database is created.

Example

- The following example creates the database `test.db` using simple obfuscation, and configures the DBA user ID and password:

```
dbinit -dba DBA,passwd -ea simple test.db
```

1.9.2.8.8 Encrypting an Existing Database (CREATE ENCRYPTED DATABASE Statement)

Create a strongly encrypted copy of a database along with encrypted copies of any associated transaction logs, transaction log mirrors, or dbspace files.

Prerequisites

By default, you must have the SERVER OPERATOR system privilege to execute the CREATE ENCRYPTED DATABASE statement. The required privileges can be changed by using the -gu database server option.

The database you are encrypting must not be running.

Context

The CREATE ENCRYPTED DATABASE statement creates a copy of the file (in this case, in encrypted form), and does not overwrite the original database file.

> ⚠ Caution
>
> For strongly encrypted databases, store a copy of the key in a safe location. If you lose the encryption key, there is no way to access the data, even with the assistance of Technical Support. The database must be discarded and you must create a new database.

Procedure

1. Start and then shut down the database that you are encrypting to ensure that the database does not require automatic recovery.

   If you cannot cleanly shut down the database, then you must use CREATE ENCRYPTED FILE statement instead of the CREATE ENCRYPTED DATABASE statement.
2. In Interactive SQL, connect to the utility database, or to any other database that is not the database that you are encrypting.

   For example, connect to the utility database:

   1. Start the database server.

      ```
      dbsrv17 -n myserver -su myuser -ID,mypassword
      ```

   2. Connect to the utility database.
      In Interactive SQL, open the *Connect* window, select *Connect with a connection string* as the *Action*, and specify the following for the connection string:

      ```
      "UID=DBA;PWD=mypassword;Server=myserver;DBN=utility_db;ASTART=No"
      ```
3. Create the encrypted copy of the database by executing the CREATE ENCRYPTED DATABASE statement.

   If you receive an error that the database cannot be started in read-only mode because it requires recovery, then you must use the CREATE ENCRYPTED FILE statement instead.

Results

A copy of the database file is created in encrypted form. If there are transaction logs, transaction log mirrors, or dbspaces associated with the database, encrypted copies of those files are made as well.

If the database uses transaction log or transaction log mirror files, then the word new is added to the beginning of the filenames. For example the transaction long `filename.log` is changed to `newfilename.log`). If the database contains dbspace files, an E (for encrypted) is added to the file extension. For example, the file `mydbspace.dbs` is changed to `mydbspace.dbsE`.

Example

The following statement takes the database file `temp.db`, and creates an AES-encrypted copy of it named `encryptedtemp.db`.

```
CREATE ENCRYPTED DATABASE 'C:\\Users\\Public\\Documents\\SQL Anywhere
          17\\Samples\\temp.db'
FROM 'C:\\Users\\Public\\Documents\\SQL Anywhere
          17\\Samples\\demo.db'
KEY 'abcpassword'
KDI '2000'
ALGORITHM 'AES';
```

Next Steps

Start the encrypted copy of the database with the encryption key, and ensure that you can connect to it.

Related Information

Comparison of the CREATE ENCRYPTED DATABASE and CREATE ENCRYPTED FILE Statements [page 1701]
CREATE ENCRYPTED DATABASE Statement

1.9.2.8.9 Encrypting an Existing Database (CREATE ENCRYPTED FILE Statement)

Create an encrypted copy of an existing database and its associated files when you cannot use the CREATE ENCRYPTED DATABASE statement. For example, you need to encrypt your database for support purposes and the CREATE ENCRYPTED DATABASE statement returns an error.

Prerequisites

By default, you must have the SERVER OPERATOR system privilege. The required privileges can be changed by using the -gu database server option.

The database being encrypted cannot be running.

Context

The CREATE ENCRYPTED DATABASE statement is the recommended method for creating an encrypted copy of the database. If the CREATE ENCRYPTED DATABASE statement returns errors, then use the CREATE ENCRYPTED FILE statement.

The database being encrypted cannot have table encryption enabled.

You must execute the CREATE ENCRYPTED FILE statement against the database file as well as against each of the database-related files independently (transaction log, transaction log mirror, dbspace files, if any).

Procedure

1. In Interactive SQL, connect to the utility database, or to any other database that is not the database that you are encrypting.

For example, connect to the utility database:

1. Start the database server.

   ```
   dbsrv17 -n server-name -su myuser -ID,mypassword
   ```

2. Connect to the utility database.
   In Interactive SQL, open the *Connect* window, select *Connect with a connection string* as the *Action*, and specify the following for the connection string:

   ```
   "UID=DBA;PWD=mypassword;Server=myserver;DBN=utility_db;ASTART=No"
   ```

2. Create the encrypted copy of the database by executing the CREATE ENCRYPTED FILE statement.

   For example:

   ```
   CREATE ENCRYPTED FILE 'temp2.db'
   FROM 'C:\\Users\\Public\\Documents\\SQL Anywhere
             17\\Samples\\demo.dbdemo.db'
      KEY 'Sd8f6654*Mnn'
      ALGORITHM 'AES_FIPS';
   ```

   > ⚠ Caution
   >
   > For strongly encrypted databases, store a copy of the key in a safe location. If you lose the encryption key, there is no way to access the data, even with the assistance of Technical Support. The database must be discarded and you must create a new database.

3. For each associated file, such as the transaction log, transaction log mirror, and dbspace files, execute the CREATE ENCRYPTED FILE statement and specify the same encryption key you used to encrypt the database file.

   For example:

   ```
   CREATE ENCRYPTED FILE 'temp2.log'
     FROM 'C:\\Users\\Public\\Documents\\SQL Anywhere
             17\\Samples\\demo.log'
      KEY 'Sd8f6654*Mnn'
      ALGORITHM 'AES_FIPS';
   ```

1.9.2.8.10 Decrypting a Database (SQL)

Decrypt a database along with any associated transaction log, transaction log mirror, and dbspace files.

Prerequisites

By default, you must have the SERVER OPERATOR system privilege to execute the CREATE DECRYPTED DATABASE statement. The required privileges can be changed by using the -gu database server option.

Context

The CREATE DECRYPTED DATABASE statement is the recommended method for decrypting a database. If the CREATE DECRYPTED statement returns errors, then you must use the CREATE DECRYPTED FILE statement instead.

The database being decrypted must not be running.

Procedure

1. Start and then shut down the database that you are decrypting to ensure that the database does not require automatic recovery.

   If you cannot start or cleanly shut down the database, then you must use CREATE DECRYPTED FILE statement instead of the CREATE DECRYPTED DATABASE statement.
2. In Interactive SQL, connect to the utility database, or any other database that is not the database that you are decrypting.
3. Execute a CREATE DECRYPTED DATABASE statement.

   If you receive an error that the database cannot be started in read-only mode because it requires recovery, then you must use the CREATE DECRYPTED FILE statement instead.

Results

A copy of the database file in decrypted form is created. The original database file is not overwritten. If there are transaction logs, transaction log mirrors, or dbspaces associated with the database, decrypted copies of those files are made as well.

Example

The first statement creates an AES256-encrypted copy of the `temp.db` database called `encryptedtemp.db`. The second statement creates a decrypted copy of `encryptedtemp.db` called `decryptedtemp.db`.

```
CREATE ENCRYPTED DATABASE 'C:\temp\encryptedtemp.db'
   FROM 'C:\temp\temp.db'
   KEY 'abc'
   ALGORITHM 'AES256';
CREATE DECRYPTED DATABASE 'C:\temp\decryptedtemp.db'
   FROM 'C:\temp\encryptedtemp.db'
   KEY 'abc';
```

Related Information

CREATE DECRYPTED DATABASE Statement
CREATE DECRYPTED FILE Statement

1.9.2.8.11 Performance Issues When Using Encryption

Performance is slower when the database is encrypted. The performance impact depends on how often pages are read from or written to disk, and can be minimized by ensuring that the server is using an adequate cache size.

You can increase the starting size of the cache with the -c option when you start the server. For operating systems that support dynamic resizing of the cache, the cache size that is used may be restricted by the amount of memory that is available; to increase the cache size, increase the available memory.

Related Information

Tip: Use the Cache to Improve Performance [page 1440]
-c Database Server Option [page 400]

1.9.2.9 Message Encryption Using Public and Private Keys

Encrypt, sign, and verify messages with an RSA key pair.

Message Encryption With RSA Key Pairs

To send a small block data to a remote computer when you are uncertain whether the computer or network is secure, and you must ensure that only the intended recipient can access the data, use RSA key pairs for message encryption.

Use the sp_generate_key_pair system procedure to create a key pair comprised of a private key and a public key. RSA encryption uses two mathematically related keys, one private and one public. Both keys are returned in PEM format. Larger keys can take several minutes to create, but are more secure. The length of the generated public and private keys depends on the `key_size` parameter. The length of the public key is approximately 80+s/6 bytes, and the length of the private key is approximately 110+7*s/9 bytes, where s is the key size in bits. Keys less than 2048 bits are insecure. If the provided `key_size` is too small to hold the full value, then it is not changed. The maximum size of the encrypted message is `key-size` minus 11 for PKCS1 padding and `key-size` minus 42 for OAEP padding. If PADDING=NONE, then the maximum size of the message is equal to the key size. The public key encrypts data and the corresponding private key decrypts it. Using the same key for both encryption and decryption fails.

Message Signing and Verification with RSA Key Pairs

To send a message when the privacy of the message is unimportant, but you want the recipient to know that the message is authentic and has not been modified since being written, use RSA key pairs to sign and verify the data. After creating a key pair, use the SECURE_SIGN_MESSAGE function to digitally sign the message. The recipient then uses the SECURE_VERIFY_MESSAGE function to verify the message. A message that is signed by using a private key can only be verified by using a public key. Signing data involves creating a cryptographic hash of a message and then encrypting the hash by using the sender's private key. The recipient verifies the signature by decrypting it using the sender's public key, computing the hash of the message, and comparing the two hashes. If they match, then the recipient knows that only someone with access to the sender's private key could have sent the message and also knows that the message has not been modified.

In this section:

Related Information

sp_generate_key_pair System Procedure
SECURE_SIGN_MESSAGE Function [String]
SECURE_VERIFY_MESSAGE Function [String]

1.9.2.9.1 Creating Key Pairs

Create RSA key pairs to use when encrypting messages or signing and verifying digital messages.

Context

To use functions that encrypt data using the RSA algorithm, first create an RSA private and public key pair.

Procedure

1. In Interactive SQL, execute the following statements to create variables to hold the private and public keys:

```
CREATE VARIABLE @publickey LONG VARCHAR;
```

```
CREATE VARIABLE @privatekey LONG VARCHAR;
```

2. Execute the following statement to generate a 2048-bit RSA key pair:

```
CALL sp_generate_key_pair( 2048, @publickey, @privatekey );
```

3. Save both the private and public key in a table or a file. The private key must be kept secure, while you can publish the public key or send it to anyone who needs it.

Results

An RSA key pair, consisting of a private key and a public key, has been created and saved in a database table or file.

Related Information

sp_generate_key_pair System Procedure

1.9.2.9.2 Encrypting and Decrypting Messages

Encrypt a message using RSA key pairs.

Prerequisites

The recipient of the message created an RSA key pair and sent only the public key to the message sender.

Context

Encrypting messages is useful for sending a small piece of data to a remote computer when you are not certain whether the network connection or the remote computer are secure and you want to ensure that only the recipient of the message can read the data.

In the scenario below, both the message sender and the message recipient have access to the same database.

Procedure

1. The sender encrypts the message with the public key that was created by the message's intended recipient. For example, the sender executes the following statements to encrypt the message using the recipient's public key and to insert the encrypted message and the recipient's name into the database.

   ```
   INSERT INTO DataTable ( data, recipient );
       VALUES ( ENCRYPT( 'Message text', @RecipientsPublicKey, 'RSA' ),
   'recipient-name');
   ```

   The only person who can decrypt and read the message is someone with access to the recipient's private key.
2. The message recipient decrypts the message using their private key. For example:

   ```
   SELECT DECRYPT ( data, @RecipientsPrivateKey, 'RSA' )
   AS decrypted_message
   FROM DataTable
   WHERE recipient = 'recipient-name';
   ```

   The statement returns the decrypted message.

Results

A message has been encrypted by the sender, inserted into the database, and decrypted and read by the message recipient.

Related Information

sp_generate_key_pair System Procedure
ENCRYPT Function [String]
DECRYPT Function [String]

1.9.2.9.3 Signing and Verifying Messages

Use RSA key pairs to sign and verify messages.

Prerequisites

Create an RSA key pair and publish the public key so that recipients of your message can access it.

Context

Digitally sign a message to assure recipients of its authenticity and that it has not been modified since being sent.

Procedure

1. The message sender executes the following statements to create a signature and to sign a message using the signature:

   ```
   CREATE VARIABLE @signature LONG BINARY;
   SELECT SECURE_SIGN_MESSAGE( document, private-key ) INTO @signature;
   ```

2. The message sender sends the document, along with the signature, to the message recipients.
3. The message recipients execute the following statements to verify that the message is authentic and that it has not been modified since being sent:

   ```
   CREATE VARIABLE @verified INTEGER;
   SELECT SECURE_VERIFY_MESSAGE( document, signature, senders-public-key ) INTO
   @verified;
   ```

   If the @verified variable equals 1, then the message is authentic.

Results

A signed message has been sent and its authenticity verified by its recipients.

Related Information

SECURE_SIGN_MESSAGE Function [String]
SECURE_VERIFY_MESSAGE Function [String]

1.9.3 Transport Layer Security

Transport Layer Security (TLS), an IETF standard protocol, secures client/server communications using digital certificates and public-key cryptography.

Transport layer security enables encryption, tamper detection, and certificate-based authentication.

You can use transport layer security to:

- Secure communications between the database server and client applications.

- Secure communications between the MobiLink server and MobiLink clients.
- Set up a secure SQL Anywhere web server.

Secure communication begins with an exchange of messages (a handshake) including **server authentication**.

Transport layer security uses server certificates to establish and maintain a secure connection. You create unique certificate files for each server.

You can use server authentication for SQL Anywhere client/server communication or for MobiLink synchronization:

- For client/server communication, a database client verifies the identity of a SQL Anywhere database server.
- For MobiLink synchronization, a MobiLink client (SQL Anywhere or UltraLite) verifies the identity of a MobiLink server.

FIPS-certified encryption requires a separate license.

Efficiency

The transport layer security protocol uses a combination of public-key and symmetric-key encryption. Public-key encryption provides better authentication techniques, but is computationally intensive. Once a secure connection is established, the client and server use a highly efficient symmetric cipher with 128-bit key size for the rest of their communication.

Certificates

Use the Certificate creation utility (createcert) to create X.509 certificate files for transport layer security. However, if you need to verify the existence of third-party certificates, or if you need more secure certificates, you can purchase the certificates from certificate authorities.

In this section:

SQL Anywhere Client/Server Communication Encryption [page 1745]
You can encrypt SQL Anywhere client/server communication using transport layer security.

SQL Anywhere Web Services Encryption [page 1751]
The SQL Anywhere web server supports HTTPS connections using TLS version 1.0 or later.

Related Information

Database Encryption and Decryption [page 1697]
Database Security
Separately Licensed Components

1.9.3.1 TLS Support

Transport Layer Security (TLS) encryption versions 1.0, 1.1, and 1.2 are supported.

Cipher suites

The following ciphers are supported.

- ECDHE_RSA_AES_256_GCM_SHA384
- ECDHE_RSA_AES_256_CBC_SHA384
- ECDHE_RSA_AES_128_GCM_SHA256
- RSA_AES_256_GCM_SHA384
- RSA_AES_128_GCM_SHA256
- ECDHE_RSA_AES_256_SHA
- ECDHE_RSA_AES_128_SHA
- RSA_AES_256_CBC_SHA
- RSA_AES_128_CBC_SHA

RSA Encryption

Non-certified RSA encryption is included with the software and can be used for client/server communication, synchronization, and web services. This version is not FIPS-certified.

FIPS-Certified RSA Encryption

FIPS-certified RSA encryption is available under separate license.

Related Information

Separately Licensed Components

1.9.3.2 FIPS-certified Encryption Technology

You can use FIPS-certified encryption algorithms to encrypt your database files, or to encrypt communications for database client/server communication, web services, and client/server communication.

Federal Information Processing Standard (FIPS) 140-2 specifies security requirements for encryption algorithms. The FIPS 140-2 certification program is a joint effort between the American National Institute of Standards and Technology (NIST) and the Canadian Communications Security Establishment (CSE). The Federal agencies of both Canada and the United States accept products that are certified as FIPS 104-2 conforming.

SQL Anywhere provides a 32-bit and 64-bit FIPS compliant module (the SAP cryptographic library, CommonCryptoLib) for encryption. On Microsoft Windows, you must use the 64-bit module (libraries) on a 64-bit system.

FIPS-Certified RSA Encryption

FIPS-certified encryption requires a separate license.

Enabling FIPS-Certified Encryption

You can ensure the use of FIPS-certified encryption on the database server using the -fips database server option. You can also use the -fips option with MobiLink servers.

You can select the use of FIPS-certified encryption for client/server communication using the FIPS network protocol option of the Encryption connection parameter.

Related Information

Simple Obfuscation Versus Strong Encryption [page 1698]
Supported Platforms

1.9.3.3 How to Set up Transport Layer Security

Set up transport layer security on your system.

Prerequisites

Obtain digital certificates. You need identity files and certificate files.

The server identity file contains the private key and should be stored securely with the database or MobiLink server.

You distribute the signer's (root) certificate file to your clients. The signer's certificate file is the one used to sign the server identity file.

You can buy certificates from a certificate authority or you can use the Certificate Creation utility (createcert). Functionality to create certificates, which is especially useful for development and testing, is provided.

Procedure

1. If you are setting up transport layer security for SQL Anywhere client/server applications:

   Start the SQL Anywhere database server with transport layer security.

   Use the -ec database server option to specify the type of security (TLS), the server identity file name, and the password that was used to protect the private key in the identity file.

   If you also want to allow unencrypted connections over shared memory, specify the -es option.

   Configure client applications to use transport layer security.

   Specify the path and file name of trusted certificates using the Encryption connection parameter (ENC).
2. If you are setting up transport layer security for SQL Anywhere web services:

   Start the SQL Anywhere database server with transport layer security.

   Use the -xs database server option to specify the type of security (HTTPS), the server identity file name, and the password that was used to protect the private key in the identity file.

   Configure browsers or other web clients to trust certificates.

   Encrypt SQL Anywhere web services.
3. If you are setting up transport layer security for MobiLink synchronization:

   Start the MobiLink server with transport layer security.

   Use the mlsrv17 -x option to specify the security stream, the server identity file name, and the password that was used to protect the private key in the identity file.

   Configure MobiLink clients to use transport layer security.

   Supply the appropriate security or network protocol options with the MobiLink synchronization client utility (dbmlsync) or UltraLite application. Specify the security stream and trusted server certificate file names.

Results

You have set up transport layer security on your system.

In this section:

Tutorial: Set Up Transport Layer Security (TLS) in SQL Anywhere [page 1732]
A brief tutorial to create one-way server authentication and encryption of the data moving between the client and server computers.

Related Information

Digital Certificates [page 1737]
Database Server with Transport Layer Security [page 1746]
Client Application Configuration to Use Transport Layer Security [page 1747]
SQL Anywhere Web Services Encryption [page 1751]
MobiLink Client Configuration to Use Transport Layer Security
Starting the MobiLink Server with Transport Layer Security
-xs Database Server Option [page 529]

1.9.3.3.1 Tutorial: Set Up Transport Layer Security (TLS) in SQL Anywhere

A brief tutorial to create one-way server authentication and encryption of the data moving between the client and server computers.

Context

Use TLS for communication between the client and server. A database client verifies the identity of an SQL Anywhere database server through authentication and ensures that the server it's contacting is the correct server. The server identity certificate remains on the database server computer while the database administrator sends a copy of the trusted certificate to each client to connect to the database server.

Creating Certificates

Create your own trusted root certificate file and a server identity certificate file.

Procedure

1. Generate an RSA trusted root certificate called `rsaroot.crt`. Use the `createcert` tool that is provided with SQL Anywhere software.

   ```
   > createcert -x -ca 1 -co rsaroot.crt -ko rsaroot.key -kp test1 -io dummy -
   scn "SelfSigner Certification Authority" -so "SelfSigner, Inc." -sou
   "SelfSigner Signing Department" -sl "Waterloo" -sst "ON" -sc "CA" -v 10 -u
   6,7 -b 2048 -m "101"
   ```

   The option `-u 6,7` describes the purpose of the certificate as 6 = Certificate Signing, 7 = CRL Signing to sign or revoke certificates.

   The option `-kp` determines the password to protect and encrypt the private key. The example specifies `test1` as the password for this walkthrough.

   For more information about the options, see Certificate Creation Utility (createcert) [page 1130].

   The files `rsaroot.crt` and `rsaroot.key` appear in the current directory. Both files are used in the identity file signing process.
2. Generate an RSA server identity file called `rsaserver.id`, signed by the trusted root certificate created in Step 1.

   ```
   > createcert -ca 0 -co dummy1 -ko dummy2 -kp test2 -io rsaserver.id -c
   rsaroot.crt -ck rsaroot.key -cp test1 -scn "YKFN1234" -so "SuperDuper
   Financial, Inc." -sou "SuperDuper Accounts" -sl "Cambridge" -sst "ON" -sc
   "CA" -v 10 -u 1,3 -b 2048 -m "102"
   ```

   The Common Name (CN) is specified by the option `-scn` is the server's host computer name. For the purpose of this example, `YKFN1234` is the database server host name. Substitute your database server's computer host name where `YKFN1234` is used.

   > **i Note**
   >
   > If an incorrect host name is used, a `TLS handshake failure` error occurs when you try to use the certificate.

   The identity certificate usage is `-u 1,3`, which describes the purpose of the identity certificate as 1 = Digital Signature, 3 = Key Encipherment.

   The password that protects the signer's private key is specified as `-cp test1`, and a new password to protect and encrypt the private key in the identity certificate is defined using the option `-kp test2`.

   The file `rsaserver.id` appears in the current directory.

Examining Certificates

Check the contents of the trusted root certificate and the server identity file.

Context

The contents of certificate files is text. Trusted root certificates have one section in their file, while the server identity file has two or more certificate sections in their file. Each section is bordered by the following two lines of text:

-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----

The rest of the file's content is encoded text, which can be read by the `viewcert` tool. `viewcert` is provided with SQL Anywhere software.

In our example, the server identity file should have two sections with a final section bordered by the following two lines of text:

-----BEGIN ENCRYPTED PRIVATE KEY-----

-----END ENCRYPTED PRIVATE KEY-----

Procedure

1. Use `viewcert` to list the attributes of the trusted root certificate `rsaroot.crt`.

```
> viewcert rsaroot.crt
```

```
X.509 Certificate
-----------------
Common Name:              SelfSigner Certification Authority
Country Code:             CA
State/Province:           ON
Locality:                 Waterloo
Organization:             SelfSigner, Inc.
Organizational Unit:      SelfSigner Signing Department
Issuer:                   SelfSigner Certification Authority
Serial Number:            101
Issued:                   Apr 29, 2021  17:09:00
Expires:                  Apr 30, 2031  17:09:00
Signature Algorithm:      RSA, SHA256
Key Type:                 RSA
Key Size:                 2048 bits
Basic Constraints:        Is a certificate authority, path length limit: 10
Key Usage:                Certificate Signing, CRL Signing
Subject Alternative Name: DNS:SelfSigner Certification Authority
```

2. Use `viewcert` to list the attributes of the server identity file rsaserver.id.

```
> viewcert rsaserver.id
```

```
X.509 Certificate
-----------------
Common Name:               YKFN1234
Country Code:              CA
State/Province:            ON
Locality:                  Cambridge
Organization:              SuperDuper Financial, Inc.
Organizational Unit:       SuperDuper Accounts
Issuer:                    SelfSigner Certification Authority
Serial Number:             102
Issued:                    Apr 29, 2021  17:09:00
Expires:                   Apr 30, 2031  17:09:00
Signature Algorithm:       RSA, SHA256
Key Type:                  RSA
Key Size:                  2048 bits
Basic Constraints:         Is not a certificate authority
Key Usage:                 Digital Signature, Key Encipherment
Extended Key Usage:        Server Authentication, Client Authentication
Subject Alternative Name: DNS:YKFN1234
X.509 Certificate
-----------------
Common Name:               SelfSigner Certification Authority
Country Code:              CA
State/Province:            ON
Locality:                  Waterloo
Organization:              SelfSigner, Inc.
Organizational Unit:       SelfSigner Signing Department
Issuer:                    SelfSigner Certification Authority
Serial Number:             101
Issued:                    Apr 29, 2021  17:09:00
Expires:                   Apr 30, 2031  17:09:00
Signature Algorithm:       RSA, SHA256
Key Type:                  RSA
Key Size:                  2048 bits
Basic Constraints:         Is a certificate authority, path length limit: 10
Key Usage:                 Certificate Signing, CRL Signing
Subject Alternative Name: DNS:SelfSigner Certification Authority
Private Key
-----------
Encrypted
```

There are two certificates in this file and a copy of the private key. The second certificate is a copy of the trusted root certificate `rsaroot.crt`. The private key is a copy of `rsaroot.key`, encrypted using the `-kp test2` password.

Results

In the identity file, you can see the relationship between the first certificate and the second certificate. The first certificate with Common Name = YKFN1234 is issued by Issuer = SelfSigner Certification Authority, which is the second certificate with Common Name = SelfSigner Certification Authority. This relationship is called the **chain**.

A **self-signed** certificate is one that is issued by itself. In this example, the second certificate with Common Name = SelfSigner Certification Authority was issued by SelfSigner Certification Authority. This also marks the beginning of the chain.

You can follow a relationship chain until the start or **root** of the chain. Any additional certificates in the identity file, called **intermediate** certificates, are linked together with a relationship chain. Always look at the Common Name and Issuer fields to determine the relationship. Going from the root certificate through the intermediate certificates to the end-user or identity certificate is called the SSL or TLS **certificate chain**.

Using Certificates

Use the certificates previously created to establish communication between the client and server.

Context

As a certificate signer, you distribute the trusted root certificate `rsaroot.crt` to all of your clients.

> ⚠ Caution
>
> Never give away a copy of the private key `rsaroot.key` or the server identity file that contains a copy of the key.

Procedure

1. To enable TLS on the database server, specify the database server option `-ec`. The following example uses the identity file:

   ```
   -ec "TLS(tls_type=RSA;IDENTITY=rsaserver.id;IDENTITY_PASSWORD=test2)"
   ```

   The identity file password is the password specified in Step 2 of *Creating Certificates*.
2. Once the server is started, a client computer can attempt to connect to the server. The client computer requires a copy of the trusted root certificate `rsaroot.crt`. No passwords or other certificate or key files are required.
3. To connect to the database server from the client computer, use the Encryption (ENC) connection parameter. The following example uses the Interactive SQL tool:

   ```
   dbisql -c
   "HOST=YKFN1234:2638;UID=DBA;PWD=XXXXXX;ENC=TLS(trusted_certificate=rsaroot.crt
   )"
   ```

   If a `TLS handshake failure` error occurs when you try to connect to the server, check the Common Name field of the first certificate in the server identity file. The Common Name is the name specified using the `-scn` option in Step 2 of *Creating Certificates*. The HOST name should match the Common Name field.

To check if the Common Name is causing the error, you can disable the Common Name check with the following change to your connection parameters:

```
dbisql -c
"HOST=YKFN1234:2638;UID=DBA;PWD=XXXXXX;ENC=TLS(trusted_certificate=rsaroot.crt
;skip_certificate_name_check=on)"
```

1.9.3.4 Digital Certificates

You need digital certificates to set up transport-layer security.

Obtain certificates from a certificate authority, or create them using the Certificate Creation utility (createcert).

Certificates From the Operating System Certificate Store

By default, secure connections use your computer's operating system certificate store to obtain a trusted certificate for secure connections.

For all secure connections except HTTPS on Windows operating systems, stored certificates are cached, with the cache reloading every 24 hours. If a required certificate is installed within 24 hours after the first secure connection, then the connection requiring that certificate fails until the cache is reloaded. To make the certificate accessible before 24 hours, restart the server.

On Windows, access the certificate store by running the following command at a command prompt:

```
certmgr.msc
```

Certificate Creation Utility

Use the Certificate Creation utility (createcert), to generate X.509 certificate files using RSA.

Certificate Viewer Utility

Use the Certificate Viewer utility (viewcert), to read X.509 certificates using RSA.

Certificates For Server Authentication

Obtain a certificate from a certificate authority or use the Certificate Creation utility (createcert) to create certificate files for server authentication. In each case, create an identity file and a certificate file.

For server authentication, you create a server identity file and a certificate file to distribute to clients.

Certificate Configurations

The certificate can be self-signed or signed by a commercial Certificate Authority.

Self-signed certificates

Self-signed server certificates can be used for simple setups.

Root certificates

A root certificate can be used to sign server certificates to improve data integrity and extensibility for multi-server deployments.

- You can store the private key used to sign server certificates in a secure central location.
- For server authentication, you can add MobiLink or database servers without reconfiguring clients.

Commercial Certificate Authorities

You can use a third-party Certificate Authority instead of a root certificate. Commercial Certificate Authorities have dedicated facilities to store private keys and create high-quality server certificates.

In this section:

Self-signed Root Certificates [page 1739]
Self-signed root certificates can be used for simple setups involving a single MobiLink or database server.

Certificate Chains [page 1739]
If you require multiple identity files, you can improve security and extensibility by using certificate chains instead of self-signed certificates.

Globally Signed Certificates [page 1742]
Globally signed certificates are high-quality certificates that are used to sign your certificate requests.

Related Information

Root Certificates [page 1741]
Certificate Creation Utility (createcert) [page 1130]
Certificate Viewer Utility (viewcert) [page 1138]

1.9.3.4.1 Self-signed Root Certificates

Self-signed root certificates can be used for simple setups involving a single MobiLink or database server.

> → Tip
>
> Use certificate chains or commercial certificate authorities if you require multiple server identity files. Certificate authorities provide extensibility and a higher level of certificate integrity with dedicated facilities to store root private keys.

Certificate

For server authentication certificates, the self-signed certificate is distributed to clients. It is an electronic document including identity information, the public key of the server, and a self-signed digital signature.

Identity file

For server authentication certificates, the identity file is stored securely with a MobiLink or database server. It is a combination of the self-signed certificate (that is distributed to clients) and the corresponding private key. The private key gives the MobiLink or database server the ability to decrypt messages sent by the client in the initial handshake.

Related Information

Certificate Chains [page 1739]
Server Authentication
Database Server with Transport Layer Security [page 1746]
Certificate Creation Utility (createcert) [page 1130]

1.9.3.4.2 Certificate Chains

If you require multiple identity files, you can improve security and extensibility by using certificate chains instead of self-signed certificates.

Certificate chains require a root certificate to sign identities. You can create this root certificate yourself or obtain one from a certificate authority.

Benefits of Using Certificate Chains

Certificate chains provide the following advantages:

Extensibility

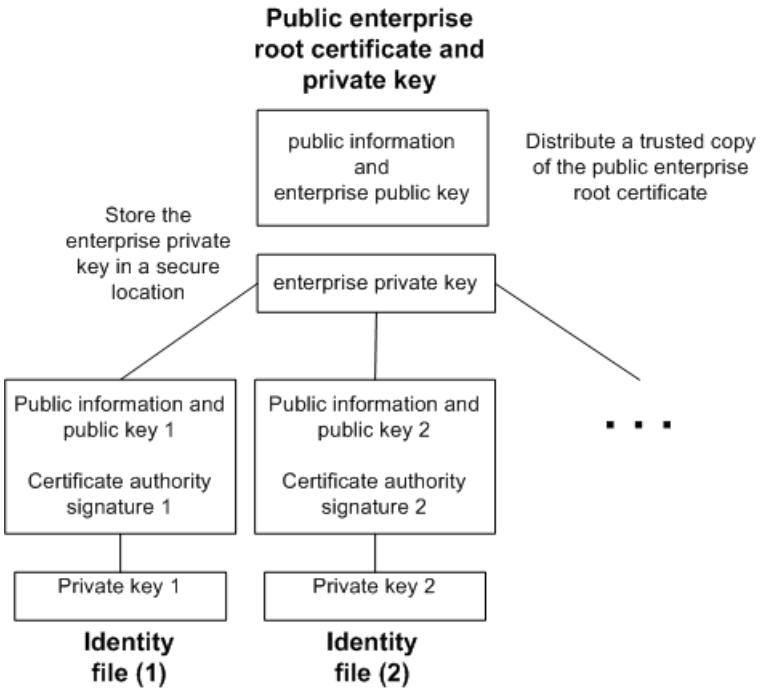
For server authentication, you can configure clients to trust any certificate signed by a specific root certificate.

If you add a new MobiLink or database server, clients do not require a copy of the new certificate.

Security

The root certificate's private key is not in the identity file. Storing the root certificate's private key in a high-security location, or using a Certificate Authority with dedicated facilities, protects the integrity of server authentication.

The following diagram provides the basic root certificate architecture.



Using Certificates in a Multi-Server Environment

To create certificates used in a multi-server environment:

- Generate a public root certificate and private key.
  Store the root private key in a secure location, preferably a dedicated facility.
  For server authentication, you distribute the public root certificate to clients.
- Use the root certificate to sign identities.
  Use the public root certificate and root private key to sign each identity. For server authentication, the identity file is used for the server.

You can also use a third-party Certificate Authority to sign your server certificates. Commercial Certificate Authorities have dedicated facilities to store private keys and create high-quality server certificates.

In this section:

Root Certificates [page 1741]
Root certificates improve data integrity and extensibility for multi-server deployments.

Signed Identity Files [page 1741]
You can use a root certificate to sign database server identity files.

Related Information

Self-signed Root Certificates [page 1739]
Globally Signed Certificates [page 1742]
Certificate Creation Utility (createcert) [page 1130]

1.9.3.4.2.1 Root Certificates

Root certificates improve data integrity and extensibility for multi-server deployments.

- You can store the private key used to create trusted certificates in a dedicated facility.
- For database server authentication, you can add database servers without reconfiguring clients.

To set up root certificates, you create the root certificate and the private key that you use to sign identities.

Related Information

Signed Identity Files [page 1741]
Globally Signed Certificates [page 1742]

1.9.3.4.2.2 Signed Identity Files

You can use a root certificate to sign database server identity files.

For database server authentication, you generate identity files for each server. Since these certificates are signed by a root certificate, you use the createcert -s option.

Related Information

Certificate Creation Utility (createcert) [page 1130]

1.9.3.4.3 Globally Signed Certificates

Globally signed certificates are high-quality certificates that are used to sign your certificate requests.

These high-quality certificates are created and managed by a commercial Certificate Authority.

Globally signed certificates have the following advantages:

- For inter-company communication, common trust in an outside, recognized authority may increase confidence in the security of the system. A Certificate Authority must guarantee the accuracy of the identification information in any certificate that it signs.
- Certificate Authorities provide controlled environments and advanced methods to generate certificates.
- The private key for the root certificate must remain private. Your organization may not have a suitable place to store this crucial information, whereas a Certificate Authority can afford to design and maintain dedicated facilities.

Setting up Globally Signed Certificates

To set up globally signed identity files, you:

- Create a certificate request using the createcert utility with the -r option.
- Use a Certificate Authority to sign each request. You can combine the signed request with the corresponding private key to create the server identity file.

> **i** Note
>
> You might be able to globally sign a root certificate. This is only applicable if your Certificate Authority generates certificates that can be used to sign other certificates.

In this section:

Globally Signed Identity Files [page 1743]
You can use globally signed certificates directly as server identity files.

Client Trust Setup for the Certificate Authority's Certificate [page 1743]
For server authentication, you must ensure that clients contacting your server trust the root certificate in the chain.

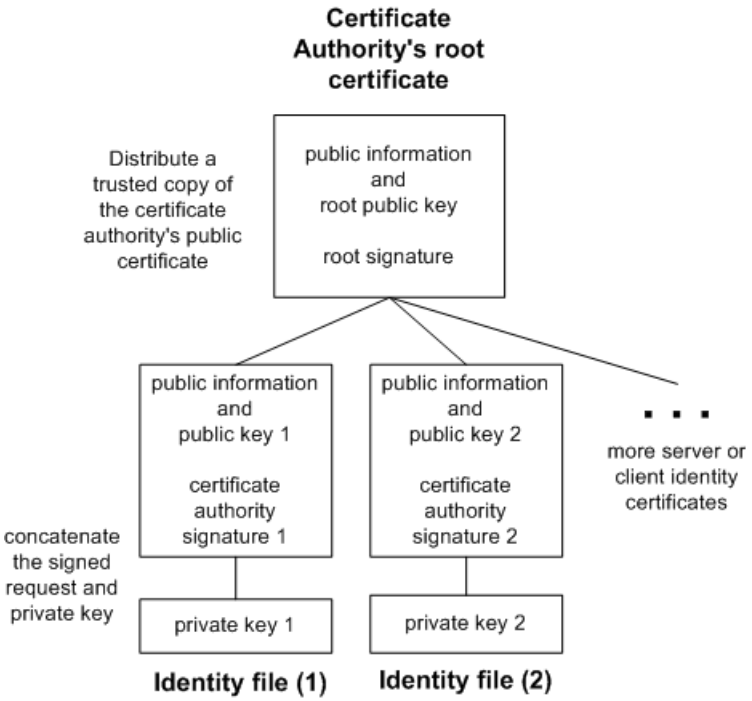
Related Information

Certificate Creation Utility (createcert) [page 1130]

1.9.3.4.3.1 Globally Signed Identity Files

You can use globally signed certificates directly as server identity files.

The following diagram shows the configuration for multiple identity files:



Reference the server identity file and the password for the private key on the dbsrv17 or mlsrv17 command line.

Related Information

Database Server with Transport Layer Security [page 1746]
Starting the MobiLink Server with Transport Layer Security

1.9.3.4.3.2 Client Trust Setup for the Certificate Authority's Certificate

For server authentication, you must ensure that clients contacting your server trust the root certificate in the chain.

For globally signed certificates, the root certificate is the Certificate Authority's certificate.

When using a globally signed certificate, each client must verify certificate field values to avoid trusting certificates that the same Certificate Authority has signed for other clients.

Related Information

MobiLink Client Configuration to Use Transport Layer Security
Database Server with Transport Layer Security [page 1746]
Server Authentication

1.9.3.5 Using Client-side Certificates for TLS

Use client-side certificates so that clients connect to your database server only if they use a certificate signed by a trusted source.

Context

This example creates the necessary certificates and assumes you are using a single root certificate to sign the client certificate and database server certificate. In a production environment you may wish to use two different root certificates and purchase database server and client certificates from a certificate authority.

Procedure

1. Use the Certificate Creation utility (createcert) to create the self-signed root certificate. For example, run the following command all on one line:

   ```
   createcert -b 2048 -x -ca 1 -co myroot.crt -io myroot.id -ko myroot.pk -kp
   root_pw -m 123 -sc CA -sst Ontario
    -sl Waterloo -so MyCompany -sou MyUnit -scn "Sample root certificate" -u 6,7
   -v 10
   ```

2. Use createcert to create the database server certificate. For example, run the following command all on one line, replacing MyServerHost with the host name of the computer that the database server runs on:

   ```
   createcert -b 2048 -c myroot.crt -ck myroot.pk -cp root_pw -ca 0 -co
   myserver.crt -io myserver.id -ko myserver.pk -kp server_pw -m 124
   -sc CA -sst Ontario -sl Waterloo -so MyCompany -sou MyUnit -scn MyServerHost -
   u 1,3,4,5 -v 10
   ```

3. Use createcert to create the client certificate. For example, run the following command all on one line:

   ```
   createcert -b 2048 -c myroot.crt -ck myroot.pk -cp root_pw -ca 0 -co
   myclient.crt -io myclient.id -ko myclient.pk -kp client_pw -m 125
    -sc CA -sst Ontario -sl Waterloo -so SAP -sou MyUnit -scn MyClientHost -u
   1,3,4,5 -v 10
   ```

   MyClientHost is arbitrary and does not need to match any host name.

4. Start the database server and include the identity and identity_password options to specify the database server's identity file and the trusted_certificate option to specify the root certificate. For example, run the following command all on one line:

```
dbsrv17 -n MyDBServer MyDatabase.db -x tcpip -ec
"TLS(identity=myserver.id;identity_password=server_pw;trusted_certificate=myro
ot.crt)"
```

   The root certificate is the certificate that signs the client's certificate.
5. Test the client application connection. For example, using the Ping utility (dbping), run the following command all on one line.

```
dbping -d -c
"UID=DBA;PWD=sql;HOST=MyServerHost;DBN=MyDatabase;ENC=TLS(trusted_certificate=
myroot.crt;identity=myclient.id;identity_password=client_pw;certificate_name=M
yServerhost)"
```

   The identity and identity_password options specify the client's identity file and the trusted_certificate option specifies the root certificate for the database server (the certificate that signs the client's certificate).

Results

You have created a trusted certificate for your client and your database server has accepted your client application connection by verifying the client certificate.

Related Information

Transport Layer Security [page 1727]
Digital Certificates [page 1737]
Certificate Creation Utility (createcert) [page 1130]
Encryption (ENC) Connection Parameter [page 80]
Ping Utility (dbping) [page 1195]

1.9.3.6 SQL Anywhere Client/Server Communication Encryption

You can encrypt SQL Anywhere client/server communication using transport layer security.

In this section:

Client Application Configuration to Use Transport Layer Security [page 1747]
You can configure SQL Anywhere client applications to use transport layer security.

Related Information

SQL Anywhere Web Services Encryption [page 1751]

1.9.3.6.1 Database Server with Transport Layer Security

To start the database server with Transport Layer Security (TLS), supply the server identity file name and the password protecting the private key.

Use the -ec database server option to specify the identity file and its password. To allow unencrypted connections over shared memory, you must also specify the -es option.

Following is the syntax of a partial dbsrv17 command line:

```
-ec  TLS(
  IDENTITY=identity-file;
  IDENTITY_PASSWORD=password )
-x tcpip
```

identity-file

The path and file name of the server identity file. If you are using FIPS-certified RSA encryption, you must generate your certificates using the RSA algorithm.

An identity file contains the public certificate and its private key. For certificates that are not self signed, the identity file also contains all the signing certificates.

password

The password for the server private key. You specify this password when you create the server certificate.

You can also start the database server with simple obfuscation. Simple obfuscation makes it difficult for someone using a packet sniffer to read the network packets sent between the client and the server, but does not assure data integrity or provide server verification.

You specify the TCP/IP protocol using the -x database server option.

Example

The following example uses the -ec database server option to specify transport layer security, the server identity file, and the password protecting the server's private key:

```
dbsrv17 -ec tls(identity=rsaserver.id;identity_password=test) -x tcpip secure.db
```

You can hide the command line options using a configuration file and the File Hiding utility (dbfhide).

Related Information

Digital Certificates [page 1737]
How to Set up Transport Layer Security [page 1731]
-fips Database Server Option [page 426]
-ec Database Server Option [page 418]
-es Database Server Option [page 424]
-x Database Server Option [page 523]
File Hiding Utility (dbfhide) [page 1160]
@data Database Server Option [page 397]

1.9.3.6.2 Client Application Configuration to Use Transport Layer Security

You can configure SQL Anywhere client applications to use transport layer security.

Using a set of encryption connection parameters, you specify trusted certificates, the type of encryption, and the network protocol.

In this section:

Server Authentication [page 1747]
Server authentication allows a remote client to verify the identity of a database server.

How to Establish a Client Connection by Using Transport Layer Security [page 1749]
To set up client applications to use transport layer security, use the Encryption [ENC] connection parameter in your connection string.

Related Information

How to Set up Transport Layer Security [page 1731]

1.9.3.6.2.1 Server Authentication

Server authentication allows a remote client to verify the identity of a database server.

Digital signatures and certificate field verification work together to achieve server authentication.

Digital Signatures

A database server certificate contains one or more digital signatures used to maintain data integrity and protect against tampering. Following are the steps used to create a digital signature:

- An algorithm performed on a certificate generates a unique value or hash.
- The hash is encrypted using a signing certificate's or Certificate Authority's private key.
- The encrypted hash, called a digital signature, is embedded in the certificate.

A digital signature can be self-signed or signed by a root certificate or Certificate Authority.

When a client application contacts a database server, and each is configured to use transport layer security, the server sends the client a copy of its certificate. The client decrypts the certificate's digital signature using the server's public key included in the certificate, calculates a new hash of the certificate, and compares the two values. If the values match, this confirms the integrity of the server's certificate.

You must generate your certificates using RSA.

Verifying a Certificate

When using a globally signed certificate, each client must verify certificate field values to avoid trusting certificates that the same Certificate Authority has signed for other clients. This is resolved by requiring your clients to test the value of fields in the identity portion of the certificate. A Certificate Authority must guarantee the accuracy of the identification information in any certificate that it signs.

You verify these fields using corresponding client connection parameters. If It is strongly recommended that you verify certificate fields if you are using a third-party Certificate Authority to globally sign certificates.

Organization

The organization field corresponds to the certificate_company encryption protocol option.

Organizational unit

The organizational unit field corresponds to the certificate_unit encryption protocol option.

Common name

The common name field corresponds to the certificate_name encryption protocol option.

Using the skip_certificate_name_check Protocol Option

This boolean protocol option controls whether the host name of the database server is checked against the host name of the database server certificate when making a TLS or HTTPS connection. Setting this option to ON is not recommended because this setting prevents the client from fully authenticating the database server. If this option is set to OFF, and if none of the certificate_name, certificate_company, or certificate_unit protocol options are set, then the host name of the database server must match the host name of the database server certificate.

If the IP address of the database server has been encoded in the database server's certificate, then you must specify the IP address, rather than the host name, in the HOST parameter.

Using the trusted_certificates Protocol Option

Clients use the trusted_certificates encryption protocol option to specify trusted database server certificates. The trusted certificate can be a server's self-signed certificate, a public root certificate, or a certificate belonging to a commercial Certificate Authority. The trusted_certificates protocol option accepts the name of a file containing one or more PEM-encoded X.509 trusted root certificates, a string of one or more PEM-encoded root certificates, or an asterisk indicating the operating system certificate store.. If the trusted_certificates protocol option is not specified, then the database server certificate is checked against the certificates in the OS certificate store.

Related Information

Self-signed Root Certificates [page 1739]
Globally Signed Certificates [page 1742]
Digital Certificates [page 1737]
Certificate Chains [page 1739]
certificate_company Protocol Option (Client Side Only) [page 195]
certificate_unit Protocol Option (Client Side Only) [page 198]
certificate_name Protocol Option (Client Side Only) [page 196]
Encryption (ENC) Connection Parameter [page 80]
trusted_certificate Protocol Option [page 244]

1.9.3.6.2.2 How to Establish a Client Connection by Using Transport Layer Security

To set up client applications to use transport layer security, use the Encryption [ENC] connection parameter in your connection string.

The connection string takes the following form.

```
{ Encryption | ENC } = TLS [ (
  [ FIPS={ ON | OFF }; ]
  [ TRUSTED_CERTIFICATE={ public-certificate-filename | * | NONE }; ]
  [ CERTIFICATE_COMPANY=organization; ]
  [ CERTIFICATE_NAME=common-name; ]
  [ CERTIFICATE_UNIT=organization-unit ]
  [ SKIP_CERTIFICATE_NAME_CHECK={ ON | OFF } ]
  [ IDENTITY=identity-file  ]
  [ IDENTITY_PASSWORD=password ]
  [ TLS_TYPE=rsa  ]
  [ DIRECT={ YES | NO } ]
  [ MIN_TLS_VERSION=ver ]
) ]
```

FIPS

Specify whether FIPS-certified encryption is to be used. FIPS-certified encryption requires a separate license.

TRUSTED_CERTIFICATE

Specify a public trusted certificate file name. Specify an asterisk (*) if the trusted certificate is in the operating system certificate store. Specify NONE to make a TLS connection without verifying the server's certificate (not recommended). If TRUSTED_CERTIFICATE is omitted, then the trusted certificate is searched in the operating system certificate store. `public-certificate-filename` is the path and file name of a file that contains one or more trusted certificates.

CERTIFICATE_COMPANY

Specify the Organization in the database server's identity file. `organization` forces the client to accept server certificates only when the Organization field on the certificate matches this value.

CERTIFICATE_NAME

Specify the Common Name in the database server's identity file. `common-name` forces the client to accept server certificates only when the Common Name field on the certificate matches this value.

CERTIFICATE_UNIT

Specify the Organizational Unit in the database server's identity file. `organization-unit` forces the client to accept server certificates only when the Organizational Unit field on the certificate matches this value.

SKIP_CERTIFICATE_NAME_CHECK Specify this option to skip the check of the database server host name (Host connection parameter) against the Common Name in the database server's identity file.

IDENTITY and IDENTITY_PASSWORD Use the IDENTITY and IDENTITY_PASSWORD option to specify an identity file and password for the client (not the database server).

TLS_TYPE This option need not be specified and is retained for backward compatibility.

DIRECT Specify DIRECT=YES when connecting through a proxy. You cannot use DIRECT=YES with the SERVER or ENG parameters, or with the LINKS parameter if you specify VERIFY=YES.

MIN_TLS_VERSION Specify the minimum downgrade TLS version. The supported values for `ver` are 1.0, 1.1, and 1.2 (the default). The period is optional, so you can also specify 10, 11, or 12.

Remarks

The keyword *TRUSTED_CERTIFICATES* is also accepted.

If any one of the CERTIFICATE_COMPANY, CERTIFICATE_NAME, CERTIFICATE_UNIT, or SKIP_CERTIFICATE_NAME_CHECK options is specified, then the check for the host name matching the Common Name in the database server's identity file is skipped. These options are useful when the database server's host computer name (HOST connection parameter) does not match the Common Name in the database server's identity file and you get a TLS handshake error.

For one-way authentication in which the client authenticates the database server, use the TRUSTED_CERTIFICATE option. If you want to enable two-way authentication in which the database server also authenticates the client, then use the IDENTITY and IDENTITY_PASSWORD option to specify an identity file and password for the client. The database server specifies the trusted certificate that was used to sign the client's identity file.

Example

The following example uses the trusted_certificates encryption connection parameter to specify the certificate, `public_cert.crt`.

```
"UID=DBA;PWD=passwd;Host=myhost;Server=myserver;
ENC=tls(trusted_certificate=public_cert.crt)"
```

The following example uses the trusted_certificates encryption connection parameter to specify the certificate, `public_cert.crt`, and verifies certificate fields using the certificate_unit and certificate_name encryption connection parameters.

```
"UID=DBA;PWD=passwd;Host=myhost;Server=myserver;
ENC=tls(trusted_certificate=public_cert.crt;
certificate_unit=test_unit;certificate_name=myserver.sample.com)"
```

Related Information

Digital Certificates [page 1737]
Server Authentication
trusted_certificate Protocol Option [page 244]
certificate_company Protocol Option (Client Side Only) [page 195]
certificate_name Protocol Option (Client Side Only) [page 196]
certificate_unit Protocol Option (Client Side Only) [page 198]
Encryption (ENC) Connection Parameter [page 80]

1.9.3.7 SQL Anywhere Web Services Encryption

The SQL Anywhere web server supports HTTPS connections using TLS version 1.0 or later.

To set up transport layer security for SQL Anywhere web services, perform the following steps:

Obtain digital certificates

You need database server certificate files and identity files. Certificates (which can be Certificate Authority certificates) are distributed to browsers or web clients. server identity files are stored securely with your SQL Anywhere web server.

Start the web server with transport layer security

Use the -xs database server option to specify HTTPS, the server identity file, and the password to protect the private key.

Configure web clients

Configure browsers or other web clients to trust certificates. The trusted certificate can be self-signed, a root, or a Certificate Authority certificate.

Related Information

Digital Certificates [page 1737]
-fips Database Server Option [page 426]
-xs Database Server Option [page 529]
identity Protocol Option [page 210]
identity_password Protocol Option [page 211]

1.9.4 Data Protection in SQL Anywhere

SQL Anywhere provides the technical enablement and infrastructure to allow you run applications on SQL Anywhere to conform to the legal requirements of data protection in the different scenarios in which SQL Anywhere is used.

Introduction to data protection

Data protection is associated with numerous legal requirements and privacy concerns. In addition to compliance with general data privacy regulation, it is necessary to consider compliance with industry-specific legislation in different countries. SAP provides specific features and functions to support compliance with regard to relevant legal requirements, including data protection. SAP does not give any advice on whether these features and functions are the best method to support company, industry, regional, or country-specific requirements. Furthermore, this information should not be taken as advice or a recommendation in regards to additional features that would be required in specific IT environments; decisions related to data protection must be made on a case-by-case basis, taking into consideration the given system landscape and the applicable legal requirements.

> **i** Note
>
> SAP does not provide legal advice in any form. SAP software supports data protection compliance by providing security features and specific data protection-relevant functions, such as simplified blocking and deletion of personal data. In many cases, compliance with applicable data protection and privacy laws will not be covered by a product feature. Definitions and other terms used in this document are not taken from a particular legal source.

Table 3: Glossary

Term	Definition
Personal data	Any information relating to an identified or identifiable natural person ("data subject"). An identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural, or social identity of that natural person.
Purpose	A legal, contractual, or in other form justified reason for the processing of **personal data**. The assumption is that any purpose has an end that is usually already defined when the purpose starts.
Blocking	A method of restricting access to data for which the primary **business purpose** has ended.
Deletion	The irreversible destruction of **personal data**.
Retention period	The period of time between the end of the last business activity involving a specific object (for example, a business partner) and the deletion of the corresponding data, subject to applicable laws. The retention period is a combination of the residence period and the blocking period.
End of purpose (EoP)	A method of identifying the point in time for a data set when the processing of **personal data** is no longer required for the primary **business purpose**. After the **EoP** has been reached, the data is **blocked** and can only be accessed by users with special authorization (for example, tax auditors).
Sensitive personal data	A category of personal data that usually includes the following type of information: • Special categories of personal data, such as data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, trade union membership, genetic data, biometric data, data concerning health or sex life or sexual orientation, or personal data concerning bank and credit accounts. • Personal data subject to professional secrecy • Personal data relating to criminal or administrative offenses • Personal data concerning insurances and bank or credit card accounts

Term	Definition
Residence period	The period of time between the end of business and the end of purpose (EoP) for a data set during which the data remains in the database and can be used in case of subsequent processes related to the original purpose. At the end of the longest configured residence period, the data is blocked or deleted. The residence period is part of the overall retention period.
Where-used check (WUC)	A process designed to ensure data integrity in the case of potential blocking of business partner data. An application's where-used check (WUC) determines if there is any dependent data for a certain business partner in the database. If dependent data exists, this means the data is still required for business activities. Therefore, the blocking of business partners referenced in the data is prevented.
Consent	The action of the data subject confirming that the usage of his or her personal data shall be allowed for a given purpose. A consent functionality allows the storage of a consent record in relation to a specific purpose and shows if a data subject has granted, withdrawn, or denied consent.

SQL Anywhere approach to data protection

Many data protection requirements depend on how the business semantics or context of the data stored and processed in SQL Anywhere are understood.

> **i** Note
>
> Using capabilities to communicate with other data sources, SQL Anywhere may also be used to process data that is stored in other systems and accessed through virtual tables.

In SQL Anywhere installations, the business semantics of data are part of the application definition and implementation. SQL Anywhere provides the features for working with technical database objects, such as tables. It is therefore the application that "knows", for example, which tables in the database contain sensitive personal data, or how business level objects, such as sales orders, are mapped to technical objects in the database. Applications built on top of SQL Anywhere need to make use of features provided by SQL Anywhere to implement compliance requirements for their specific use case.

SQL Anywhere provides a variety of security-related features to implement general security requirements that are also required for data protection and privacy:

Aspect of data protection and privacy	SQL Anywhere feature	More information
Access control	Several features in SQL Anywhere provide access control: • Authentication • Authorization • Data encryption (for example, data volume encryption, redo log encryption, backup encryption)	• User security (roles and privileges) [page 1526] • Data security [page 1676]
Access logging	Audit logging **⚠ Caution** Database audit logs may potentially expose personal data.	Database activity audits [page 1692]
Transmission control/communication security	Support for encrypted communication on all internal and external channels	Transport Layer Security [page 1727]
Availability control	Several features in SQL Anywhere provide availability control: • Backup and recovery • High availability • System replication (SQL Remote and MobiLink)	• Database backup and recovery [page 939] • High availability and read-only scale-out systems [page 1756]
Separation by purpose	Separation by purpose is subject to the organizational model implemented and must be applied as part of the authorization concept. Isolated data storage can be achieved in SQL Anywhere using: • Database schemas protected using authorization	Multiple databases running on a single database server [page 323]

⚠ Caution

The extent to which data protection is ensured depends on secure system operation. Network security, adequate logging of system changes, and appropriate usage of the system are the basic technical requirements for compliance with data privacy legislation and other legislation

In this section:

Deletion of Personal Data [page 1756]

SQL Anywhere supports the deletion of data in tables using SQL deletion commands. Applications running on SQL Anywhere must make use of such commands to implement deletion requirements of personal data.

1.9.4.1 Deletion of Personal Data

SQL Anywhere supports the deletion of data in tables using SQL deletion commands. Applications running on SQL Anywhere must make use of such commands to implement deletion requirements of personal data.

End of purpose checks, including implementation of legally required retention periods and data blocking, are managed by the application. Applications can implement data blocking using SQL Anywhere mechanisms such as authorization and table creation. For example, an application could transfer blocked data to separate database tables that are protected by special authorizations.

Once data has been deleted, the delete operation cannot be undone using SQL statements.

> **i** Note
>
> Following standard practice, deletion of personal data is not enforced in backups. Common practice is that deleted data disappears from backups following typical backup-rotation mechanisms.

1.10 High Availability and Read-only Scale-out Systems

Database mirroring and Veritas Cluster Server are supported to provide high availability.

Both of these configurations provide alternate database servers that can run a database should the database server currently running the database become unavailable.

To distribute workloads, consider whether using read-only scale-out is an appropriate solution.

You can also use high availability and/or read-only scale-out for hosting web applications.

In this section:

Related Information

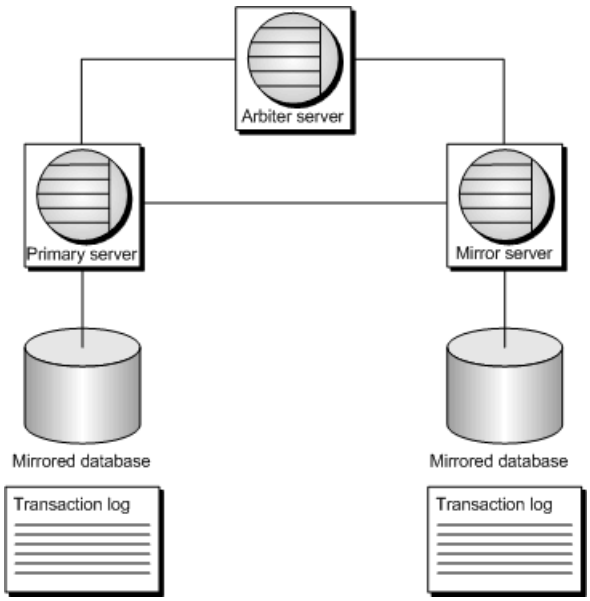
Read-only Scale-out [page 1830]
Web Services High Availability and Scale-out Solutions with Relay Server and the Outbound Enabler [page 1834]

1.10.1 Database Mirroring

Database mirroring is a configuration of three database servers, running on separate computers, that co-operate to maintain copies of the database and transaction log files.

The expected configuration for a database mirroring system is to have three database servers running on three independent physical computers so that the system can be constantly running. Database mirroring does not support suspending one of the computers.

The **primary server** and **mirror server** each maintain a copy of the database files and transaction log files. The primary server has the read-write copy of the database, while the mirror server has the read-only copy of the database. The third server, called the **arbiter server**, is used when it is necessary to determine which of the other two servers can take ownership of the database. Ownership of the database refers to the server that has the read-write copy of the database. The arbiter does not maintain a copy of the database. The configuration of three database servers (the primary, mirror, and arbiter servers) is called a **mirroring system**, and the primary and mirror servers together are called the **partner** servers.



Clients connect to the primary server to access the database. Any changes that are made to the database are recorded in the transaction log on the primary server. When the changes are committed, the transaction log pages are sent to the mirror server where they are applied to a mirror copy of the database. The copy of the database on the mirror server can only be accessed in read-only mode while that server is acting as the mirror server.

Database mirroring is a separately licensed component.

In this section:

Arbiter Server [page 1758]
The arbiter server resolves disputes between the servers regarding which server should be the primary server.

Quorum in Database Mirroring [page 1759]

Before a server can assume the role of primary server, it must have **quorum**, which means that at least one other server must agree that a server can own the database.

Database Mirroring and MobiLink [page 1760]
Configure MobiLink synchronization to run in a high availability environment using a SQL Anywhere consolidated database with SQL Anywhere remote databases.

Application Development Considerations with Database Mirroring [page 1761]
When you are using database mirroring, in almost all cases, applications should be able to run in the same manner as they do when connected to a non-mirrored database.

Setting up a Database Mirroring System [page 1771]
Set up a database mirroring system by defining primary, mirror, and arbiter servers to maintain copies of the database and transaction log files.

Database Mirroring System Maintenance [page 1777]
It is important to maintain a database mirroring system.

Troubleshooting: Database Mirroring Systems [page 1789]
There are three extended database properties that you can use to help troubleshoot your mirroring system.

Tutorial: Creating a Database Mirroring System [page 1800]
Create a database mirroring system and respond to a failover.

Tutorial: Creating a Database Mirroring System with Multiple Databases That Share an Arbiter Server [page 1808]
Create a mirroring configuration in which the primary and mirror servers each host three individual databases participating in mirroring systems.

Tutorial: Moving the Arbiter Server [page 1815]
Move an arbiter server without stopping the mirroring system by creating new server and changing the arbiter server definitions to use the new arbiter server.

Tutorial: Moving a Partner Server [page 1817]
Move a server in a database mirroring system to a different server without stopping the system, delete the mirror definitions of the current mirror server, create a new server, and add the new server to the mirroring system.

Related Information

Separately Licensed Components
Upgrades and Rebuilds in a Database Mirroring System
Connecting to a Database in a Mirroring System [page 1762]

1.10.1.1 Arbiter Server

The arbiter server resolves disputes between the servers regarding which server should be the primary server.

Without an arbiter, if server A starts in a mirroring system when server B is unavailable, server A cannot determine if its copy of the database files is the most current. Starting a database using files that are not

current results in the loss of transactions that have already been applied and committed to the other copy of the database. In addition, the other copy of the database would be unusable for mirroring once the two partner servers re-establish communication.

In addition to resolving disputes at startup, the arbiter is involved if the communication link between two servers is broken, but both of those servers are still running. Without an arbiter, both servers could assume that they should take ownership of a database. Again, this would result in lost transactions and incompatible databases. With an arbiter, the primary server can verify that it still owns the database and can remain available to clients. If the primary server loses communications with both the mirror and the arbiter, it must shut down and wait for either one to become available.

The server that runs as an arbiter server can function as arbiter for more than one mirror system. It can also act as a database server for other databases.

Related Information

Using a copy node as an arbiter [page 1776]

1.10.1.2 Quorum in Database Mirroring

Before a server can assume the role of primary server, it must have **quorum**, which means that at least one other server must agree that a server can own the database.

If the mirror server becomes unavailable while the primary server and arbiter are connected, the primary server continues to provide access to the database. If the primary server loses quorum, it can no longer permit access to the database. At that point, the primary server stops the mirrored database, attempts to restart it, and then waits to regain quorum before making the database available.

What Happens When a Database Mirroring System Starts

When you start a database mirroring system, the database servers go through a startup process to reach quorum and accept client connections. The following steps describe a typical sequence of events for this process. Assume that Server 1 and Server 2 are partners that were both stopped (Server 2 was the mirror and stopped first, and then Server 1 was stopped), and that the arbiter server is running.

1. The arbiter server waits for connections from Server 1 and Server 2.
2. Server 1 starts and tries to connect to Server 2 and the arbiter server.
3. Server 1 connects to the arbiter server.
4. Server 1 fails to connect to Server 2 because it is not running.
5. Server 1 negotiates with the arbiter to determine if Server 1 should be the primary or mirror server.
6. Quorum is reached when the arbiter and Server 1 agree that Server 1 should assume the role of the primary server. The arbiter and the state file for the Server 1 indicate that Server 1 was the most recent primary server before the partner servers stopped.
7. Server 1 starts accepting connections.

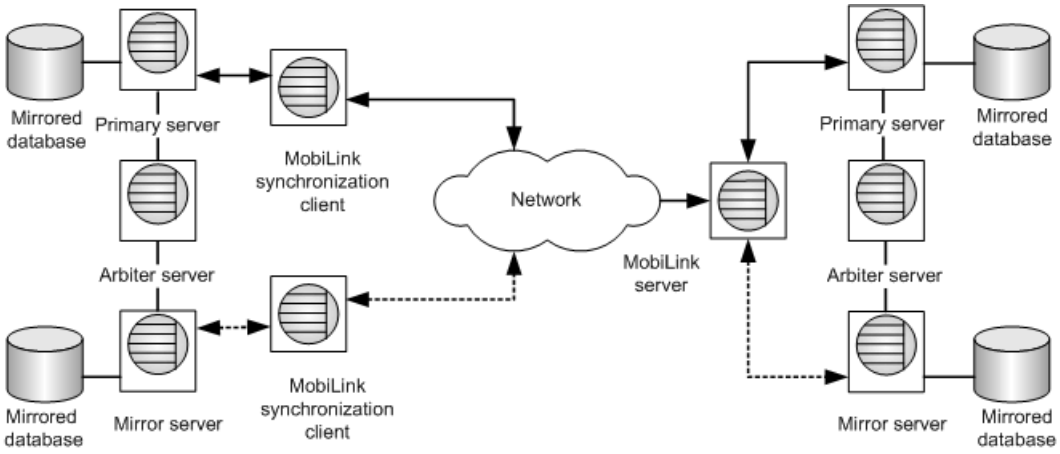
8. Server 2 starts and tries to connect to Server 1 and the arbiter.
9. Server 2 connects to the arbiter and to server 1.
10. Server 2 negotiates with Server 1 and the arbiter to determine if Server 2 should be the primary or mirror server.
11. All three servers agree Server 2 should be the mirror.
12. Server 2 starts accepting read-only connections.
13. Server 2 starts applying log file pages received from Server 1.

1.10.1.3 Database Mirroring and MobiLink

Configure MobiLink synchronization to run in a high availability environment using a SQL Anywhere consolidated database with SQL Anywhere remote databases.

You can configure your MobiLink synchronization system to run the consolidated database in a high availability environment, or the remote database to run in a high availability environment, or both.

The following graphic shows MobiLink synchronization running in an environment where both the consolidated and remote databases are mirrored:



High Availability for SQL Anywhere Consolidated Databases

The **primary server** maintains the active copy of the database file and transaction logs. This server accepts active connections and processes database transactions. All transactions are sent to the mirror server by the primary server.

The **mirror server** receives all transactions that have been applied to the primary server directly from the primary server. The mirror server becomes the primary server if failover occurs.

The **MobiLink server** establishes an ODBC connection with the primary server and processes the upload stream from the MobiLink client. It generates the download stream to be sent to the clients. The MobiLink server always connects to the server acting as primary, never to the mirror server.

High Availability for SQL Anywhere Remote Databases

Data synchronization is initiated when the **MobiLink client** establishes a connection with the remote database and the MobiLink server. The MobiLink client must scan the active transaction log and offline transaction logs. However, in a high availability environment it is not always known which database is acting as the primary server and where the active and offline transaction logs exist. If offline transaction logs are associated with the database, you must specify the OfflineDirectory extended option (for example, `-e dir=path-name`) when starting the MobiLink client.

The MobiLink client generates the upload stream from the remote database's transaction log to send to the MobiLink server. The client then processes and applies the download stream.

Recovering From Primary Server Failure

To prevent your high availability data synchronization system from having a single point of failure,yo u use multiple MobiLink servers and load balancers. In this case, each component of the data synchronization environment is redundant and the system functions even if hardware or software failures occur in the environment.

Similarly, to avoid a single point of failure at the remote database, an implementation using an external procedure, such as the xp_cmdshell system procedure, to call the MobiLink client can be triggered via a scheduled event that is stored in both the primary and mirror databases.

Related Information

MobiLink Server in a Server Farm
High Availability and Read-only Scale-out Systems [page 1756]
OfflineDirectory (dir) Extended Option
xp_cmdshell System Procedure

1.10.1.4 Application Development Considerations with Database Mirroring

When you are using database mirroring, in almost all cases, applications should be able to run in the same manner as they do when connected to a non-mirrored database.

However, there are a few considerations to take into account when developing applications that are used with database mirroring.

In this section:

Connect to the database on either the primary or mirror server by specifying the alternate name for the respective server with the ServerName connection parameter, along with the addresses for both partner servers with the Host connection parameter.

Requirements and Restrictions for Database Mirroring Systems [page 1766]
There are no special hardware or software requirements for database mirroring, and the database servers can be running in separate geographical locations.

Performance Considerations with Database Mirroring Systems [page 1767]
There are several performance considerations for database mirroring systems.

Database Mirroring Modes [page 1768]
There are three operational modes that control when and how transactions are recorded to the mirror server.

1.10.1.4.1 Connecting to a Database in a Mirroring System

Connect to the database on either the primary or mirror server by specifying the alternate name for the respective server with the ServerName connection parameter, along with the addresses for both partner servers with the Host connection parameter.

Prerequisites

The mirroring server partners must be running.

Context

In a mirroring system, you do not necessarily know which database server is acting as the primary server and which one is acting as the mirror server.

Create clients that can reconnect to the database (for example, when failover occurs the user may need to shut down the application and then restart it).

Procedure

To connect to the primary or mirror server running the mirrored database, the connection string must include the following connection parameters:

Server

To connect to the primary server

Choose one of the following options:

Option	Action
ServerName connection parameter	Specify the alternate server name for the primary database server in the database mirroring system. This is the name that is defined by the `CREATE MIRROR SERVER primary_alternate_server_name AS PRIMARY ...` statement.
NodeType connection parameter	Set the NodeType (Node) connection parameter to PRIMARY in the connection string.

To connect to the mirror server

Choose one of the following options

Option	Action
ServerName connection parameter	Specify the alternate server name for the mirror database server in the database mirroring system. This is the name that is defined by the `CREATE MIRROR SERVER mirror_alternate_server_name AS MIRROR ...` statement.
NodeType connection parameter	Set the NodeType (Node) connection parameter to MIRROR in the connection string.

HOST

Specify the addresses and ports for both partners to ensure that the connection succeeds regardless of which partner is currently the primary. The host and port information for each of the partners is defined in the connection string of the `CREATE MIRROR SERVER partner_server_name AS PARTNER connection_string='SERVER=partner_server_name;host=host_name:port_number` statements.

You might want to specify the RetryConnectionTimeout connection parameter to control how long clients keep retrying the connection attempt to either database server.

Results

The client connects to the specified server

Example

For example, if the primary database server is named myprimary, clients specify the connection parameter Server=myprimary in their connection string:

```
...UID=user12;PWD=x92H4pY;Server=myprimary;HOST=myhost1:6871,myhost2:6872...
```

In this section:

Related Information

Connecting to a Database in a Read-only Scale-out System [page 1838]
RetryConnectionTimeout (RetryConnTO) Connection Parameter [page 109]
NodeType (NODE) Connection Parameter [page 101]

1.10.1.4.1.1 Queries Executed on the Mirror Database

In a database mirroring system, you can access the database running on the mirror server by using a read-only connection.

This functionality is useful to offload reporting or other operations that require read-only access to this database.

Any attempt to change the mirror database results in an error, which is the same behavior as when a database is started as read-only using the -r option. You can perform operations on temporary tables.

Queries that are executed against the mirror database can place locks, depending on the isolation level specified. If locks interfere with operations being applied from the primary server, then the connections holding the locks have their transactions rolled back and any open cursors for those connections are closed. Applications running at isolation level 0 do not add row locks, but still acquire schema locks. If the schema locks interfere with operations being applied from the primary server, the transaction on the mirror database is just rolled back.

Applications that require a consistent view of the database (and so cannot use isolation level 0) should consider using snapshot isolation. To do so, the allow_snapshot_isolation option must be set to On. This option takes effect on both the primary server and the mirror server, so the costs associated with snapshot isolation must be considered.

Connections to the mirror database are affected by transactions against the primary server, since those operations are then processed and applied by the mirror server. There can be a small delay between the time

an update on the primary server is committed and the time that the update is available on the mirror server. Normally this delay is short, but keep this in mind when you are accessing the database running on the mirror server.

Connections to the mirror database are maintained if failover occurs and the mirror server becomes the primary server. After failover, a connection can make changes to the database. Query the value of the ReadOnly database property to determine whether the database you are connected to is updatable:

```
SELECT DB_PROPERTY( 'ReadOnly' );
```

Connections on a copy-node or mirror database are dropped or canceled in some cases when these connections prevent the transaction log from being applied. For example, if a connection is using an event that the transaction log is trying to alter or drop, then the connection that is blocking the transaction log from being applied is dropped, and a message is printed to the server console.

Related Information

Snapshot Isolation
Event Execution in a Database Mirroring or Read-only Scale-out System [page 1765]
allow_snapshot_isolation Option [page 675]
CREATE MIRROR SERVER Statement
ALTER MIRROR SERVER Statement

1.10.1.4.1.2 Event Execution in a Database Mirroring or Read-only Scale-out System

Events can execute either in a database mirroring or read-only scale-out system.

Events can execute either on only the primary or root server, or they can execute on the primary server, the mirror server and the copy nodes.

By default, events are not fired on the mirror database; event firing only starts after a failover from the primary server to the mirror server takes place.

You specify where an event can execute when you create or alter the event. Use the FOR clause of the CREATE EVENT or ALTER EVENT statement, to specify where an event can fire.

DatabaseStart Event Type

By default an event that uses the DatabaseStart event type only executes when a server becomes the primary server for the database. However, if you have specified that the event can run on the primary server, mirror server, and the copy nodes, then the event is executed when any database starts. If in a mirroring system this event did not run when the database started (for example, the database was running before the event was created), then the event can execute during a failover.

For example if you start a database mirroring system, create a DatabaseStart event with the FOR ALL clause, and then stop the primary server, you cause a failover. In this example, the event executes on the new primary server. The DatabaseStart event does not execute during subsequent failovers.

Related Information

System Events [page 1001]
Ping Utility (dbping) [page 1195]
EVENT_PARAMETER Function [System]

1.10.1.4.2 Requirements and Restrictions for Database Mirroring Systems

There are no special hardware or software requirements for database mirroring, and the database servers can be running in separate geographical locations.

Database servers that are participating in a database mirroring system can run both mirrored and non-mirrored databases. As well, the arbiter server can be the arbiter for multiple database mirroring systems.

If a database server is running more than one mirrored database, each mirrored database should be in a separate directory.

The following requirements and restrictions apply when using database mirroring:

Network database server required

Because mirroring involves network communication between the database servers, you must use the network database server (dbsrv17); the personal database server cannot be used.

All database servers in a database mirroring system must use the same minor release; however, they can run different Support Packages.

TCP/IP required

Only TCP/IP connections are permitted between mirroring servers.

Transaction log restrictions

You cannot truncate the transaction log on the primary when you are using database mirroring because the truncation can result in lost transactions. You can rename the transaction log as often as necessary.

Web servers

If you are using SQL Anywhere as a web server and in a mirroring system, it is not possible to specify a URL for a web request in a way that guarantees that the request is directed to the current primary server. If one of the server computers is named in the URL and that server is down, the request times out.

Regular back ups are still required

Database mirroring is not a replacement for a backup and recovery plan. Implement a backup and recovery strategy for your database.

DDL restrictions

Minimize the number of DDL statements executed on the primary database while an object could be in use on a copy node or the mirror server. Dropping or altering a system object may cause user connections to the mirror or copy nodes to be dropped if those connections are using the object that was dropped or altered.

Failover and scheduled events

If your database has scheduled events and failover occurs, the failover must complete before the events start; otherwise, the events are not executed until their next scheduled time. If the mirror server is changing to the primary server role but has not completed the transition, the scheduled event is not executed; it is executed at the next scheduled time. When an event is running and the primary server loses its connections to the mirror and arbiter servers, the event connection and all other connections are dropped. If an event is scheduled to run after the mirror assumes the primary server role, the event runs on the new primary server.

Related Information

Transaction Log File Management in a Database Mirroring System [page 1775]
Backups in a Database Mirroring System [page 1778]
Database Backup and Recovery [page 939]
Transaction Log File Management in a Database Mirroring System [page 1775]
Queries Executed on the Mirror Database [page 1764]
Web Services High Availability and Scale-out Solutions with Relay Server and the Outbound Enabler [page 1834]
Troubleshooting: Using Database Mirroring System Events to Send Notification Email When Failover Occurs [page 1780]

1.10.1.4.3 Performance Considerations with Database Mirroring Systems

There are several performance considerations for database mirroring systems.

- The computers running the primary and mirror servers should be configured with similar hardware (processor, disk, memory, and so on). At any given time, the database server running on either computer can be acting as the primary server for the database being mirrored. The mirror server utilization is typically low, depending on update activity on the primary and the load generated by any read-only connections to the mirror server.
- Transactions on the mirror server may lag behind those on the primary server in being applied. Longer mirror lag time causes read-only connections to the mirror to access further out-of-date information, and may cause a longer recovery time if the mirror server is restarted or must take over as primary. Use the lagtime mirror option to control the acceptable level of lag between the primary and mirror servers. The primary server reduces its rate of transactions when the lag time approaches the value of this setting. If the lag time exceeds the value of the setting, warnings are written to the transaction log until it is no longer running behind.
- The performance of transactions that update the database depends on the size of the transaction and the frequency of commits. Increasing either the size of the transactions or the frequency of the commits

causes the primary server to interact more with the mirror server, creating more opportunities for the primary server to be delayed and for the user to notice the delays.

- The performance of queries against the primary server is usually not affected by mirroring. If your application does more reads than updates and auditing is not enabled, you are unlikely to experience a change in the performance of the primary server. But, if database auditing is enabled, you may experience a change in performance because auditing writes additional information to the transaction log.
- The performance of your client applications can be affected by the network connection between the servers in the mirroring system, especially when the servers are located in different geographic locations. A slow network connection between the servers can degrade the performance of your client applications.
- A mirroring system running in either asynchronous or asyncfullpage mode requires the primary server to wait in fewer cases than when running in synchronous mode. But, even in asynchronous or asyncfullpage mode, some of the messages sent from the primary server to the mirror server wait for an acknowledgement from the mirror, potentially introducing delays. In addition, there is a small amount of overhead in sending packets to the mirror. Asynchronous and asyncfullpage modes are not recommended because transactions can be lost.
- Incomplete transactions must be rolled back when the mirror server takes ownership of the database, and the longer a transaction is, the longer it takes to roll the transaction back. The recovery speed for failover is affected by the number of clients and the length of their transactions that must be rolled back. If recovery speed is a concern, you can design your application to use short transactions whenever possible.

Related Information

Database Mirroring Modes [page 1768]

1.10.1.4.4 Database Mirroring Modes

There are three operational modes that control when and how transactions are recorded to the mirror server.

- synchronous (default)
- asynchronous
- asyncfullpage

In almost all cases, the default mode, synchronous, is recommended as it is the most reliable.

These modes control when and how transactions are recorded on the mirror server, and you set them by using the SET MIRROR OPTION statement to set the value of the synchronization_mode option.

The sending of log pages to the copy nodes is always done asynchronously, regardless of the mode chosen.

When choosing a synchronization mode for your database mirroring system, you must determine whether the performance of modifications to the primary database or avoiding the loss of transactions is more important when failover occurs.

You can check the database mirroring mode by querying the value of the MirrorMode database property:

```
SELECT DB_PROPERTY( 'MirrorMode' );
```

Synchronous Mode (Default)

In synchronous mode, committed transactions are guaranteed to be recorded on the mirror server. Should a failure occur on the primary server, no committed transactions are lost when the mirror server takes over. In this mode, the primary server sends transaction log pages to the mirror when a transaction is committed. The mirror server acknowledges that transmission when it has written those pages to its copy of the transaction log. The primary server does not reply to the application until it receives this acknowledgement.

Using synchronous mode provides **transaction safety** because the partner servers are in a synchronized state, and changes sent to the mirror must be acknowledged before the primary can proceed.

Asynchronous and Asyncfullpage Modes (Not Recommended)

Asynchronous and asyncfullpage mode are faster than synchronous mode, but are less reliable. By default, if the primary server stops unexpectedly, then the mirror server does not automatically take over as the primary.

Asynchronous mode

In asynchronous mode, committed transactions are not guaranteed to be recorded on the mirror server. In this mode, the primary server sends transaction log pages to the mirror when a transaction is committed. It does not wait for an acknowledgement from the mirror before replying to the application that the COMMIT has completed. Should a failure occur on the primary server, some committed transactions may be lost when the mirror server takes over.

Asyncfullpage mode

In asyncfullpage (or page) mode, pages are not sent on COMMIT; instead, they are sent when the page is full. This setting reduces the amount of traffic between the two database servers and improves the performance of the primary server. If the current log page has not been sent to the mirror for the number of seconds specified by the pagetimeout parameter, it is sent even though it is not yet full. The default pagetimeout is 5 seconds. Using this mode provides a limit on how long committed transactions are exposed to being lost if the primary server goes down and the mirror server takes ownership of the database. Asyncfullpage mode implies asynchronous operation, so the primary server does not wait for an acknowledgement from the mirror.

In asynchronous or asyncfullpage mode, failover from the primary server to the mirror server is not automatic because the mirror server may not have all committed transactions that were applied on the primary server. For this reason, when using one of the asynchronous modes, a mirror server, by default, cannot take ownership of a database when the primary fails. If automatic failover is desirable in this situation (despite the likelihood of lost transactions), set the auto_failover option to on using the SET MIRROR OPTION statement. Otherwise, when the failed server is restarted, it detects whether transactions were lost. If transactions were lost, it writes a message to the database server message log and shuts down the database. The current database and transaction log must then be replaced using a backup before mirroring can continue.

> **i** Note
>
> Set the auto_failover mirroring option to ON if you are using asynchronous or asyncfullpage mode. Then, if the primary server goes down, the mirror server automatically takes over as the primary server. Using the auto_failover option can result in lost transactions.

The synchronize_mirror_on_commit database option controls when database changes are guaranteed to have been sent to a mirror server when running in asynchronous or asyncfullpage mode. When you set this option to

On, each COMMIT causes any changes recorded in the transaction log to be sent to the mirror server and an acknowledgement to be sent by the mirror server to the primary server once the changes are received by the mirror server. The option can be set for specific transactions by using SET TEMPORARY OPTION. It may also be useful to set the option for specific applications by examining the APPINFO string in a login procedure.

SQL Anywhere supports system events that fire when failover occurs in a database mirroring system, regardless of which mode you are using. You can use these events for such tasks as notifying the administrator when failover occurs.

When running in asynchronous or asyncfullpage mode, you must determine what happens when failover occurs and transactions are not committed to the database.

A mirroring system running in either asynchronous or asyncfullpage mode requires the primary server to wait in fewer cases than when running in synchronous mode. But, even in asynchronous or asyncfullpage mode, some of the messages sent from the primary server to the mirror server wait for an acknowledgment from the mirror, potentially introducing delays. In addition, there is a small amount of overhead in sending packets to the mirror. Asynchronous and asyncfullpage modes are not recommended because transactions can be lost.

In this section:

Related Information

Troubleshooting: How to Recover from Primary Server Failure [page 1791]
Troubleshooting: How to Recover from Primary Server Failure [page 1791]
synchronize_mirror_on_commit Option [page 843]
SET OPTION Statement
SET MIRROR OPTION Statement
Troubleshooting: Using Database Mirroring System Events to Send Notification Email When Failover Occurs [page 1780]

1.10.1.4.4.1 Mirror Synchronization States

When a mirroring system is using synchronous mode, it can be in one of two states: synchronizing or synchronized.

synchronizing

The mirror server is not connected or has not yet read all the primary server's log pages. This value is also returned if the synchronization mode `is` asynchronous

synchronized

The mirror server is connected and has all changes that have been committed on the primary server.

Once a partner server starts and determines that it is acting as the mirror, it first requests any log pages from the primary server that it does not already have. This may involve copying pages from log files other than the

current active log on the primary server. As it receives these pages, the mirror applies the changes they contain to its copy of the database. Once all pages from the primary have been received, the primary and mirror are in a synchronized state. From that point onward, any changes committed on the primary must be sent to the mirror and acknowledged by the mirror.

In asynchronous and asyncfullpage mode, the mirror requests log pages as above; however, the two servers never enter a synchronized state. Once the mirror has requested all log pages available at the primary, the primary is notified that it must send any updated pages to the mirror.

In a read-only scale-out system, the synchronization state of copy nodes is always synchronizing.

1.10.1.5 Setting up a Database Mirroring System

Set up a database mirroring system by defining primary, mirror, and arbiter servers to maintain copies of the database and transaction log files.

Prerequisites

You must have MANAGE ANY MIRROR SERVER system privilege.

This task involves backing up a copy of the database. Depending upon the backup method you choose, see the appropriate privileges for that method.

Context

In a production mirroring system, the partners and arbiter servers must run on three separate computers. When setting up a database mirroring system, in all the examples below localhost and the port number must be changed to the computer name and port where the corresponding database server must run.

Procedure

1. Start the database to be mirrored on a database server with the *-su* and *-xp on* options. This server becomes one of the partners for the database, the initial primary server. The database must have a transaction log. For example:

   ```
   dbsrv17 -n mirror_server1 -x tcpip(PORT=6871;DOBROAD=no) -su passwd "c:
   \server1\mirror_demo.db" -xp on
   ```

2. Connect to the database. For example:

   ```
   dbisql -c "UID=DBA;PWD=passwd;SERVER=mirror_server1"
   ```

3. Define the partner servers and arbiter server for the database by using the CREATE MIRROR SERVER statement. Also define the primary and mirror servers as partners in the database mirroring system.

For example:

- The following two SQL statements define mirror_server1 and mirror_server2 as the partner servers in the database mirroring system:

```
CREATE MIRROR SERVER mirror_server1
AS PARTNER
connection_string='SERVER=mirror_server1;host=localhost:6871'
state_file='c:\\server1\\server1.state';
CREATE MIRROR SERVER mirror_server2
AS PARTNER
connection_string='SERVER=mirror_server2;host=localhost:6872'
state_file='c:\\server2\\server2.state';
```

- The following SQL statements define:
  - mirror_demo_primary as the alternate server name for mirror_server1. Clients use mirror_demo_primary as the server name to connect to the database server that is acting as the primary server.
  - mirror_demo_mirror as the alternate server name for mirror_server2. Clients use mirror_demo_mirror as the server name to connect to the database server that is acting as the mirror server.

```
CREATE MIRROR SERVER mirror_demo_primary
AS PRIMARY
connection_string='SERVER=mirror_demo_primary;HOST=localhost:
6871,localhost:6872';
CREATE MIRROR SERVER mirror_demo_mirror
AS MIRROR
connection_string='SERVER=mirror_demo_mirror;HOST=localhost:6871,localhost:
6872';
```

- The following SQL statement defines the arbiter server for the database mirroring system:

```
CREATE MIRROR SERVER demo_arbiter
AS ARBITER
connection_string ='SERVER=demo_arbiter;HOST=localhost:6870';
```

4. Set mirroring options for the mirroring system. You must specify an authentication string. For example:

```
SET MIRROR OPTION authentication_string='abc';
```

5. Create a copy of the primary database and the current transaction log, as well as any other transaction logs, onto the computer where the second partner is. For example, use the BACKUP DATABASE statement.

   The transaction log files on the primary server computer and the mirror server computer must be identical, including the starting offset of the current transaction log files.

6. Start the second database server in the database mirroring system:

```
dbsrv17 -n mirror_server2 -x tcpip(PORT=6872;DOBROAD=no) -su passwd "c:
\server2\mirror_demo.db" -xp on
```

7. Start the arbiter server:

```
dbsrv17 -n demo_arbiter -su passwd -x "TCPIP(PORT=6870;DOBROAD=no)" -xf "c:
\arbiter\arbiter.state" -xa "AUTH=abc;DBN=mirror_demo"
```

Results

Clients can now connect to the mirrored database.

The roles of primary and mirror are necessary for configuring the database servers in the system: the names that you give these servers are used as alternate server names when clients connect to the database servers. Either partner server can act as the primary or mirror server.

Next Steps

Monitor your database mirroring system by adding your system to the Monitor's resource list.

Check the status of the database servers in a database mirroring system by connecting to the primary database from SQL Central. Database mirroring information is available on the *Health and Statistics* pane.

In this section:

Related Information

Database Health and Statistics [page 1109]
Upgrades and Rebuilds in a Database Mirroring System
Lesson 7: (Optional) Monitoring a Database Mirroring System [page 1367]
Stopping a Database Server in a Mirroring System (dbstop Utility) [page 1781]
Troubleshooting: The Primary Server Cannot be Restarted [page 1792]
-su Database Server Option [page 502]
-xp Database Option [page 563]
CREATE MIRROR SERVER Statement
SET MIRROR OPTION Statement
BACKUP DATABASE Statement
Backup Utility (dbbackup) [page 1121]

1.10.1.5.1 How the Primary Server Is Chosen by Database Mirroring System

During a normal startup, the following inputs affect which server becomes the primary server:

- the contents of the state information files
- the transaction log position on each database server
- the designation of a preferred primary server

If a database with no mirroring definitions is started on a database server (for example, S1) started with *-xp on*, and the mirroring definitions are then made, S1 would be the initial primary server for the database.

Alternatively, when you first set up a database mirroring system and there are no state information files, and the copies of the database and transaction log are identical, both servers are eligible to act as the primary. In this situation, the server names are compared, and the server with the lower name acts as primary. For example, the name server1 is lower than server2, so server1 becomes the primary server.

For the initial startup, both servers must be running and connected for them to agree on their roles; the presence of an arbiter is not enough since the prior state information recorded in the state information files does not exist.

In this section:

Related Information

Troubleshooting: State Information Files of the Partners and Arbiter [page 1790]
Quorum in Database Mirroring [page 1759]

1.10.1.5.1.1 Preferred Database Server in a Database Mirroring System

In a database mirroring system, you can specify one partner server as the preferred server.

The **preferred server** is the partner that by default runs as the primary server and has ownership of the database.

If the preferred server becomes unavailable, then the server that was acting as the mirror server becomes the primary server. When the preferred server restarts it:

1. Obtains any transaction log entries it does not already have from the current primary server.
2. Asks the current primary server to relinquish ownership of the database. The servers then switch roles. The preferred server becomes the primary server and the other server becomes the mirror server. Any connections to the database on the non-preferred server are lost when the database ownership changes.

You specify a preferred server by adding PREFERRED='YES' to the CREATE MIRROR SERVER statement that defines the partner server.

Example

The following statement creates the mirror_server1 server as the preferred server.

```
CREATE MIRROR SERVER mirror_server1
AS PARTNER
connection_string='SERVER=mirror_server1;host=localhost:6871'
state_file='c:\\server1\\server1.state'
preferred='YES';
```

Related Information

User-initiated Role Switches (Failovers) [page 1780]
Quorum in Database Mirroring [page 1759]
CREATE MIRROR SERVER Statement
ALTER MIRROR SERVER Statement

1.10.1.5.2 Transaction Log File Management in a Database Mirroring System

When a partner server starts, it examines all the transaction log files in the same directory as the current transaction log file and determines which ones must be applied.

The database server then applies the operations in these transaction logs to the database before determining whether to act as the primary or mirror server. You can store the transaction log files and the database file in the same directory. However, this directory should not contain other files because it can cause delays in starting the database. In particular, the directory containing the transaction log files for a database should not contain transaction log files for other databases.

The server that takes on the primary server role must have a transaction log with the same starting offset as the current transaction log on the mirror server, as well as any subsequent transaction log files up to the current transaction log file for the primary server.

Once a server takes on the mirror server role, it starts receiving transaction log pages from the primary server. When a transaction log rename occurs on the primary, the rename is also performed on the mirror. The mirror then writes new transaction log pages to a new file with the name specified for the transaction log.

Transaction log files can be deleted periodically on the primary. Each time a transaction log file is renamed, the mirror is notified about which transaction log file is the oldest surviving file on the primary. Any transaction log files older than this are deleted on the mirror.

You cannot truncate the current transaction log on the primary when you are using database mirroring because this may result in lost transactions. You can rename the transaction log as often as necessary. To

remove old transaction logs from the primary, use a scheduled event to delete them once you are certain that they are no longer needed. For example, you can create an event that runs each day and renames the transaction log. This event could also delete copies of the transaction log that are more than a week old.

Related Information

Troubleshooting: Database Mirroring Systems [page 1789]

1.10.1.5.3 Using a copy node as an arbiter

Configure one server to act as both a copy node and an arbiter for the system when you do not want to create a new database server to run the arbiter of a mirroring system that includes a read-only scale-out system.

Prerequisites

You must have the MANAGE ANY MIRROR SERVER system privilege.

This task involves backing up a copy of the database. Depending upon the backup method you choose, see the appropriate privileges for that method.

Procedure

1. Stop the copy node that you want to also be an arbiter server.
2. Choose one of the following options. Note that the command line method (dbsrv**XX** or dbeng**XX**) must be entered all on one line, even if the example is on multiple lines:

Option	Action
Add the arbiter to an existing copy node	1. Start the copy node with the -xf, -su, and -xa database server options. For example: `dbsrv17 -n scaleout_child_demo -su passwd -x "tcpip(PORT=6873)" -xf "c:\scaleoutdemo\copynode\arbiter.state" -xa "AUTH=abc;DBN=scaleoutdemo" "c:\scaleoutdemo\copynode\scaleoutdemo.db" -xp on` 2. Create the arbiter server definition so that its name does not match the server name of any of the database servers in the mirroring system. Execute a CREATE MIRROR SERVER statement. For example: `CREATE MIRROR SERVER myarbiter AS ARBITER`

Option	Action
	`connection_string = 'SERVER=scaleout_child_demo;HOST=localhost:6873';`
Add a copy node to the existing arbiter	1. Stop the existing arbiter server. 2. Connect to the primary server and delete the existing arbiter mirror server definition. 3. From the primary server, create a new arbiter mirror server definition with a different name (but with the actual server name in the connection_string). For example: `CREATE MIRROR SERVER newarbiter` `AS ARBITER` `connection_string = 'SERVER=actual_server_name;HOST=arbiter_host:6870';` 4. Make a backup of the mirrored database and copy it to computer where the copy node is to run. 5. Restart the arbiter server but include `mirrored.db -xp ON` in the server command, where mirrored.db is the name of the mirrored database.

Results

The arbiter server also runs a scale-out copy node.

Example

For example, there is no database server named TheArbiter in the mirroring and read-only scale-out system that uses the following server definition.

```
CREATE MIRROR SERVER "scaleout_child"
AS COPY
connection_string = 'SERVER=scaleout_child;HOST=winxp-2:6878';
CREATE MIRROR SERVER "TheArbiter"
AS ARBITER
connection_string = 'SERVER=scaleout_child;HOST=winxp-2:6878';
```

Related Information

CREATE MIRROR SERVER Statement

1.10.1.6 Database Mirroring System Maintenance

It is important to maintain a database mirroring system.

In this section:

1.10.1.6.1 Backups in a Database Mirroring System

Although database mirroring can help minimize the risk of data loss, it is strongly recommended that you back up and validate the databases that are participating in database mirroring systems.

- You can use the BACKUP DATABASE statement to perform a backup relative to the database server. The BACKUP DATABASE statement is executed on the primary database server, so the file name that is provided should specify a network drive or UNC name that is consistent for both the primary and mirror database servers.
- You can perform client-side backups of the database using the dbbackup utility.

Related Information

Troubleshooting: How to Recover from Primary Server Failure [page 1791]
BACKUP DATABASE Statement
Backup Utility (dbbackup) [page 1121]

1.10.1.6.2 Role Switches (Failovers) in Database Mirroring

If the primary server becomes unavailable because of hardware or software failure, the mirror server negotiates with the arbiter to take ownership of the database and assume the role of primary server.

For an ownership transfer, or **role switch**, to take place, the surviving partner server and the arbiter must agree that the mirror was in a current, synchronized state at the time the role switch is attempted.

- Any clients that were connected to the original primary server are disconnected, and any uncommitted transactions are lost. Clients must then reconnect to the database on the new primary server to continue accessing the database. When the original primary server becomes available again, it assumes the role of mirror server.
- Any clients that were connected to the original mirror database are maintained when failover occurs and the mirror server becomes the primary server. After failover, these client connections can make changes to the database.

The database servers display status messages in the database server messages window on startup to indicate which role the server is assuming and how far the startup process has progressed. A message appears if the database must be restarted because of the loss of one or more of the other servers in the mirroring system, or if its role changes from mirror to primary.

If an assertion failure occurs on a server that is part of a mirroring system, the server writes the error to the database server message log and then exits. This notifies the other servers that it has failed so that they can take appropriate action.

If a database involved in a high availability environment encounters a problem, such as an incompatible or mismatched transaction log, it is stopped. The database server running the problem database also shuts down unless there are other databases running on it.

Details about the state of each database in the database mirroring system are stored in a state information file.

- When an arbiter is present, failover from primary to mirror is automatic. If you are running in synchronous mode, no committed transactions are lost during failover.
- Failover is very fast because the mirror server has already applied the transaction log. When the mirror detects that the primary has failed, it rolls back any uncommitted transactions and then makes the database available.

If the primary becomes unavailable while it is in a state of synchronizing, then the mirror cannot take over as the primary. The mirroring system waits for the primary to become available.

System events that fire when failover occurs in a database mirroring system, regardless of which mode you are using, are supported. You can use these events for such tasks as notifying the administrator when failover occurs.

In this section:

User-initiated Role Switches (Failovers) [page 1780]
Initiate a database mirroring failover (role switch) from the primary server to the mirror server without stopping the server, by connecting to the primary server.

Troubleshooting: Using Database Mirroring System Events to Send Notification Email When Failover Occurs [page 1780]
Use MirrorServerDisconnect and the MirrorFailover events to send email notification that an action may be required on the mirror database.

Related Information

Troubleshooting: State Information Files of the Partners and Arbiter [page 1790]

1.10.1.6.2.1 User-initiated Role Switches (Failovers)

Initiate a database mirroring failover (role switch) from the primary server to the mirror server without stopping the server, by connecting to the primary server.

Then, execute the following statement:

```
ALTER DATABASE SET PARTNER FAILOVER;
```

When this statement is executed, any existing connections to the database are closed, including the connection that executed the statement. If the statement is contained in a procedure or event, other statements that follow it may not be executed. The privileges required to execute this statement are controlled by the -gd server option.

This statement is an alternative to specifying a preferred server, and can be used with logic that controls when ownership of the database is given to a specific database server. For example, you can initiate failover based on the availability of the partner server (determined by the value of the PartnerState database property), or the number of connections to the database (determined by the value of the ConnCount database property).

Related Information

Preferred Database Server in a Database Mirroring System [page 1774]
ALTER DATABASE Statement

1.10.1.6.2.2 Troubleshooting: Using Database Mirroring System Events to Send Notification Email When Failover Occurs

Use MirrorServerDisconnect and the MirrorFailover events to send email notification that an action may be required on the mirror database.

These events may not fire in all situations that result in the database running on the primary server becoming unavailable. For example, a power outage affecting both the primary and mirror servers would prevent either of these events from being fired. If this type of monitoring is required, it can be implemented on a separate computer via a scripting language by calling dbping to periodically connect to the mirror database.

Example

The following example creates an event that notifies an administrator when failover occurs:

```
CREATE EVENT mirror_server_unavailable
TYPE  MirrorServerDisconnect
HANDLER
BEGIN
 CALL xp_startmail ( mail_user ='George Smith',
                     mail_password ='mypwd' );
 CALL xp_sendmail( recipient='DBAdmin',
    subject='Mirror server disconnect occurred',
    "message"='The following server is unavailable in the mirroring system: '
    || event_parameter( 'MirrorServerName' ) );
 CALL xp_stopmail ( );
END;
```

Related Information

Ping Utility (dbping) [page 1195]
EVENT_PARAMETER Function [System]

1.10.1.6.3 Stopping a Database Server in a Mirroring System (dbstop Utility)

Stop a database in a mirroring system with the utility database.

Prerequisites

The database server must have been started using the -su database server option to set the password for the utility database. You must have the password for the utility database.

Context

Stopping a database is useful when you must stop a database server in a mirroring system to apply a Support Package or minor release, or to remove a server from the mirroring system.

Procedure

Connect to the utility database and run the dbstop utility. You must specify the connection parameters for the utility database in the connection string.

For example, the following command stops a database server named mirror_server1:

```
dbstop -c "UID=DBA;PWD=sql;DBN=utility_db;LINKS=tcpip" mirror_server1
```

Results

The database server stops. Provided that at least two servers in the high availability system remain, users should be able to connect to either copy of the database. However, depending upon which server is stopped, users may or may not be able to update the database.

Next Steps

When you stop a primary, mirror, or arbiter server, your system continues to operate; however, to maintain high availability, you must restart or replace the stopped database server.

Related Information

How to Start the Database Server [page 324]
Upgrades and Rebuilds in a Database Mirroring System
Connecting to the Utility Database (Connect Window) [page 311]
-su Database Server Option [page 502]

1.10.1.6.4 Dropping a Mirror Server from a Mirroring System

Drop a mirror server from a mirroring system by deleting the two mirror server definitions (CREATE MIRROR SERVER...AS PARTNER and CREATE MIRROR SERVER...AS MIRROR).

Prerequisites

Do not drop a mirror server that is running as part of a database mirroring system.

You can only drop the mirror server. If the server you want to drop is the primary server, you must initiate a failover so that the mirror and primary servers switch roles.

You must have the MANAGE ANY MIRROR SERVER system privilege.

Context

To keep the same server name but change its settings, you can use the CREATE OR REPLACE MIRROR SERVER statement or the ALTER MIRROR SERVER statement.

Procedure

1. Connect to the database on the primary server.
2. If the mirror server has any child copy nodes, then execute an ALTER MIRROR SERVER statement to re-assign any child copy nodes to a different parent.
3. Delete the mirror server definition by executing a DROP MIRROR SERVER statement and specifying the mirror server name:

   ```
   DROP MIRROR SERVER mirror_server_name;
   ```

4. Delete the partner server definition by executing a DROP MIRROR SERVER statement and specifying the partner name:

   ```
   DROP MIRROR SERVER partner_server_name;
   ```

5. (Optional) Stop the database server.

Results

The mirror server is dropped. If there are any copy nodes remaining, the database mirroring system becomes a read-only scale-out system.

Example

The partner servers of a mirroring system were created with the following statements:

```
CREATE MIRROR SERVER mirror_server1
AS PARTNER
connection_string='SERVER=mirror_server1;host=localhost:6871'
state_file='c:\\server1\\server1.state';
CREATE MIRROR SERVER mirror_server2
AS PARTNER
connection_string='SERVER=mirror_server2;host=localhost:6872'
state_file='c:\\server2\\server2.state';

CREATE MIRROR SERVER myprimary
AS PRIMARY
```

```
connection_string='SERVER=myprimary;HOST=localhost:6871,localhost:6872';
CREATE MIRROR SERVER mymirror
AS MIRROR
connection_string='SERVER=mymirror;HOST=localhost:6871,localhost:6872';
```

Execute the following statement to drop the mirror server (mirror_server2):

```
DROP MIRROR SERVER mirror_server2;
DROP MIRROR SERVER mymirror
```

Related Information

DROP MIRROR SERVER Statement
CREATE MIRROR SERVER Statement
ALTER MIRROR SERVER Statement

1.10.1.6.5 Adding a Mirrored Database to a Running Mirroring System

Add a mirrored database to a running mirroring system without reducing the availability of the existing servers.

Prerequisites

The primary and mirror servers must have been started using the -su database server option to set the password for the utility database. You must have the password for the utility database.

This task involves backing up a copy of the database. Depending upon the backup method you choose, see the appropriate privileges for that method.

You must also have the MANAGE MIRROR SERVER privilege.

The arbiter and one partner of the existing mirroring system must be running.

The database that you are adding must have a transaction log.

Context

You have a mirroring system running and you want to use the existing partner and arbiter servers to mirror another database.

There should be only one mirrored database with its associated log files per directory. If a server is running multiple mirrored databases, each database should be in its own directory.

Procedure

1. Ensure that the arbiter server was started using the:
   - -xa database server option with the DBN parameter. Specify DBN=* to allow the arbiter server to accept connections for any mirrored database. Otherwise, specify all the databases that must be dynamically started and use this server as their arbiter.
   - -su database server option. You must be able to connect to the utility database on the arbiter server.

   For example:

   ```
   dbsrv17 -n mirror_arbiter -su passwd -x "TCPIP(PORT=6870;DOBROAD=no)" -xf "c:
   \arbiter\arbiter.state" -xa "AUTH=abc;DBN=*" -o c:\arbiter\arbiter.conslog
   ```
2. Copy the database that you want to mirror, along with its transaction logs, to one of the existing partner computers. The additional database should be in a new directory. For example, `C:\server1\second`.
3. Connect to the utility database on the existing partner server. For example:

   ```
   dbisql -c "UID=dba;PWD=sql;DBN=utility_db;SERVER=mirror_server1"
   ```
4. Start the database to be added using the START DATABASE statement with the MIRROR ON and AUTOSTOP OFF clauses. For example:

   ```
   START DATABASE 'c:\\server1\\second\\second.db'
   AUTOSTOP OFF
   MIRROR ON;
   ```

   The database starts.
5. Connect to the database. For example:

   ```
   dbisql -c UID=DBA;PWD=sql;SERVER=mirror_server1;DBN=second
   ```

   Create the mirror server definitions (PRIMARY, MIRROR, ARBITER and both PARTNERS) and the authentication string with the SET MIRROR OPTION.

   The authentication string for the new mirroring system must match the authentication string specified by the arbiter's -xa option. The hostnames and ports must match those for the existing partner and arbiter servers. The PARTNER mirror server names must match the -n server option for the partner servers. The PRIMARY and MIRROR server names must not match any server names currently in use by the system.

   For example:

   ```
   CREATE MIRROR SERVER second_primary
   AS PRIMARY
   connection_string='SERVER=second_primary;HOST=localhost:6871,localhost:6872';
   CREATE MIRROR SERVER second_mirror
   AS MIRROR
   connection_string='SERVER=second_mirror;HOST=localhost:6871,localhost:6872';
   CREATE MIRROR SERVER mirror_server1
   AS PARTNER
   connection_string='SERVER=mirror_server1;host=localhost:6871'
   state_file='c:\\server1\\server1.state';
   CREATE MIRROR SERVER mirror_server2
   AS PARTNER
   connection_string='SERVER=mirror_server2;host=localhost:6872'
   state_file='c:\\server2\\server2.state';
   CREATE MIRROR SERVER arbiter_server
   AS ARBITER
   connection_string='SERVER=mirror_arbiter;HOST=localhost:6870';
   ```

```
SET MIRROR OPTION authentication_string='abc';
```

6. Create a backup of the database and copy the back up to the second partner server. For example, use the dbbackup utility.

```
dbbackup -c "UID=DBA;PWD=sql;SERVER=mirror_server1;DBN=second" "c:
\server2\second"
```

7. Connect to the utility database on the second partner server. For example:

```
dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_server2;DBN=utility_db"
```

8. Start the database on the second partner server using the START DATABASE statement with the AUTOSTOP OFF MIRROR ON clause.

```
START DATABASE 'c:\\server2\\second\\second.db'
AUTOSTOP OFF
MIRROR ON;
```

Results

The newly added mirrored database should be running on both partners. Confirm that one partner has the role of Primary, and the other partner has the role of Mirror (either using the console log or `DB_PROPERTY( 'MirrorRole' )`).

Related Information

-xa Database Server Option [page 525]

1.10.1.6.6 Moving a Partner Server

Move a server in a database mirroring system to a different server without stopping the system by deleting the mirror definitions of the current mirror server, creating a new server, and adding the new server to the mirroring system.

Prerequisites

You must have the MANAGE ANY MIRROR SERVER privilege.

This task involves backing up a copy of the database. Depending upon the backup method you choose, see the appropriate privileges for that method.

Context

Replace localhost with an actual computer name when trying out an example.

Procedure

1. Connect to the partner server that you want to move and ensure that it has the mirror role. You can only move the partner with the mirror role. If the server you want to move to is the primary server, you must initiate a failover so that the primary and mirror servers switch roles
2. Create a new directory for the new partner server.
3. Connect to the primary server.
4. Drop the partner definition for the server being moved by executing a DROP MIRROR SERVER statement.

   The mirror database stops. If the mirror database is the only database running on the server, then the server also stops.
5. Create a new partner definition for the server to become the new partner. For example, execute the following statement:

   ```
   CREATE MIRROR SERVER mirror_server3 AS PARTNER
   connection_string='SERVER=demo_server3;HOST=localhost:6874'
   state_file='c:\\server3\\server3.state';
   ```
6. Update the primary and mirror definitions. For example, execute the following statements:

   ```
   ALTER MIRROR SERVER mirror_demo_primary AS PRIMARY
   connection_string='SERVER=mirror_demo_primary;HOST=localhost:6871,localhost:
   6874';
   ALTER MIRROR SERVER mirror_demo_mirror AS MIRROR
   connection_string='SERVER=mirror_demo_mirror;HOST=localhost:6871,localhost:
   6874';
   ```
7. Make copies of the primary database file and transaction log and add them to the new partner server directory. For example, run the following command:

   ```
   dbbackup -c "UID=DBA;PWD=sql;SERVER=mirror_server1;DBN=mirror_demo" server3
   ```
8. Start the new partner server with the -xp option so that the new partner can join the mirroring system. For example, run the following command:

   ```
   dbsrv17 -n mirror_server3 -x "tcpip(PORT=6874)" -su passwd "c:
   \server3\mirror_demo.db" -xp on
   ```
9. Connect to the new partner server and verify that it is the mirror server.

Results

The mirror system is running with the new partner server.

Next Steps

Ensure that clients connecting to the primary server or mirror server have their connection strings updated to specify the addresses of both partners in the Host connection parameter.

Related Information

Tutorial: Moving a Partner Server [page 1817]

1.10.1.6.7 Moving the Arbiter Server

Move an arbiter server without stopping the mirroring system by creating a new server and changing the arbiter mirror server definitions to use the new arbiter server.

Prerequisites

You must have the MANAGE ANY MIRROR SERVER system privilege.

Procedure

1. Start the server to become the arbiter with the -su, -xa, and -xf options. For example:

   ```
   dbsrv17 -n demo_arbiter2 -x "tcpip(port=6873)" -xf c:\arbiter2\arbiter2.state
   -xa "AUTH=abc;DBN=mirror_demo" -su passwd
   ```

2. Connect to the primary server and alter the arbiter mirror server definition for the mirroring system so that the connection_string is for the new arbiter server.

   For example, execute the following statement to change the arbiter server definition to that of the new server.

   ```
   ALTER MIRROR SERVER demo_arbiter
   AS ARBITER
   connection_string='SERVER=demo_arbiter2;HOST=localhost:6873';
   ```

   The primary and mirror servers disconnect from the arbiter server and connect to the new arbiter server.
3. Wait a few seconds, and then stop the old arbiter server.

Results

The arbiter server is moved to the new server.

Next Steps

Ping the ArbiterState database property of the mirroring system to ensure that the new arbiter server is connected to the mirroring system. For example, run the following command:

```
dbping -c "UID=DBA;PWD=sql;SERVER=mirror_demo_primary" -pd ArbiterState
```

The value for the ArbiterState is connected:

```
    SQL Anywhere Server Ping Utility Version 16.0.4157
    Type       Property                Value
    ---------  ----------------        ------------------------------
    Database   ArbiterState            connected
```

Related Information

Tutorial: Moving the Arbiter Server [page 1815]

1.10.1.7 Troubleshooting: Database Mirroring Systems

There are three extended database properties that you can use to help troubleshoot your mirroring system.

- MirrorState
- PartnerState
- ArbiterState

In this section:

Several documented scenarios can help you understand what happens when a server becomes unavailable in a database mirroring system.

Troubleshooting: Database Mirroring Dropped Connections [page 1796]
In a database mirroring system, when connections between database servers drop, several considerations apply.

1.10.1.7.1 Troubleshooting: State Information Files of the Partners and Arbiter

The partners and the arbiter in the high availability system each maintain a state information file that records that server view of the state of the mirroring system.

The state information file is used during startup when determining the role to be assumed by a server. The server local state is compared against that of the other servers in the database mirroring system.

> ⚠ Caution
>
> Do not modify the state information files.

You must always specify a state information file for each server in the mirroring system using the state_file option of the CREATE MIRROR SERVER statement.

The state information file contains the following information:

Field	Description
Owner	Indicates which database server is the primary server.
State	Contains the synchronization state (one of synchronizing or synchronized) to indicate whether the server is receiving log pages or is up to date.
Mode	Specifies the synchronization mode (one of synchronous, asynchronous, or page).
Sequence	Contains a value indicating how many times failover has occurred on the database mirroring system. The sequence number is incremented on each role switch. It helps to determine whether a server view of the state of the mirroring system is current.

The following shows sample contents for a state information file:

```
[demo]
Owner=server2
State=synchronizing
Mode=asynchronous
Sequence=35
```

If a state information file does not exist, it is created automatically. State information files should only be modified by the database server. However, when you are resetting or re-deploying a mirror system, delete the state files.

Related Information

Mirror Synchronization States [page 1770]
Database Mirroring Modes [page 1768]
Database Mirroring [page 1757]
CREATE MIRROR SERVER Statement

1.10.1.7.2 Troubleshooting: How to Recover from Primary Server Failure

The steps for recovering from primary server failure depend on the synchronization mode you are using for your database mirroring system.

Synchronous Mode

If you are running in synchronous mode, then all the transactions that are present on the primary server are also guaranteed to be committed on the mirror server. The mirror server can take over as the new primary server without any user intervention in almost all cases.

> **i** Note
>
> In rare cases, if a primary server loses its connections to both the mirror and arbiter servers while a checkpoint is being performed, then synchronization may fail the next time the two servers connect. The mirror server reports an error indicating that there is a transaction log mismatch with the primary server and the mirror database stops. If this occurs, the database on the current primary server must be manually copied or backed up to the mirror server for the mirror server to successfully start and synchronize.

Asynchronous or Asyncfullpage Mode

In asynchronous or asyncfullpage mode, failover from the primary server to the mirror server is not automatic because the mirror server may not have all committed transactions that were applied on the primary server. Unless you specified that autofailover should take place, when using one of the asynchronous modes, a mirror server, by default, cannot take ownership of a database when the primary fails. When the failed server is restarted, it detects whether transactions were lost. If transactions were lost, it writes a message to the database server message log and shuts down the database.

When starting the original mirror server as the new primary server, you have two options for getting the database files on both servers into the same state:

- Copy the database and transaction log files from the original primary server to the mirror server and then start the mirror server as the new primary server. You can force a server to be the primary server using the ALTER DATABASE statement with the FORCE START clause.

- Perform a backup (using dbbackup) on the original mirror server. Copy the files to the original primary server, and then start the database servers.

Related Information

Database Mirroring Modes [page 1768]
Backup Utility (dbbackup) [page 1121]
ALTER DATABASE Statement

1.10.1.7.3 Troubleshooting: The Primary Server Cannot be Restarted

Force the mirror server to take over as the primary server in situations where all other attempts to restart have failed.

Prerequisites

You must have the privileges specified by the -gd database server option (SERVER OPERATOR system privilege, by default).

The partner servers must have been started with the -su database option, so that you can connect to the utility database on the mirror server.

You have already tried manually stopping the primary server and restarting it and executing an ALTER DATABASE statement with the SET PARTNER FAILOVER clause.

Context

This method uses the ALTER DATABASE statement with the FORCE START clause and can result in losing committed transactions.

You can only use the ALTER DATABASE FORCE START statement when the following conditions apply:

- The primary server is down (for example, due to a hardware failure).
- The mirror server failed to take over as the primary server.
- You can connect to the utility database on the mirror database server.

> ⚠ Caution
>
> Using the FORCE START clause can result in the loss of transactions if the primary server contains transactions that the mirror server does not have.

It is recommended that you restart the primary and execute ALTER DATABASE with the SET PARTNER FAILOVER clause to force a failure without lost transactions. The FORCE START clause should only be used when the primary cannot be restarted as a last resort.

Procedure

1. Stop the mirror database, and if possible the mirror server.
2. If the database and transaction log files from the primary server are available, back them up and validate the backed up copies. Otherwise back up and validate the transaction log.
3. If the database file and the transaction log file from the primary server are valid, then copy both files to the mirror server.

   If the database file from the primary is invalid, but the primary transaction log file is valid, then:

   a. Apply the primary log file to a backup of the mirror database and validate the database.
   b. Copy the database and transaction log file to the mirror server.
4. Start the database on the mirror server (or restart the mirror server) with mirroring enabled. If you are restarting the mirror server, specify the -su database option, so that you can connect to the utility database.
5. Connect to the utility database, utility_db, on the mirror server.
6. Execute an ALTER DATABASE FORCE START statement to force the mirror server to become the primary server.

Results

The mirror server becomes the new primary server

Example

The following statement forces the mirror server for the database `mymirroreddb.db` to become the primary server.

```
ALTER DATABASE mymirroreddb FORCE START;
```

Next Steps

To replace the failed primary machine with a different primary machine, then the failed partner must dropped, and a new partner added for the new machine. The primary and mirror connection strings must be updated.

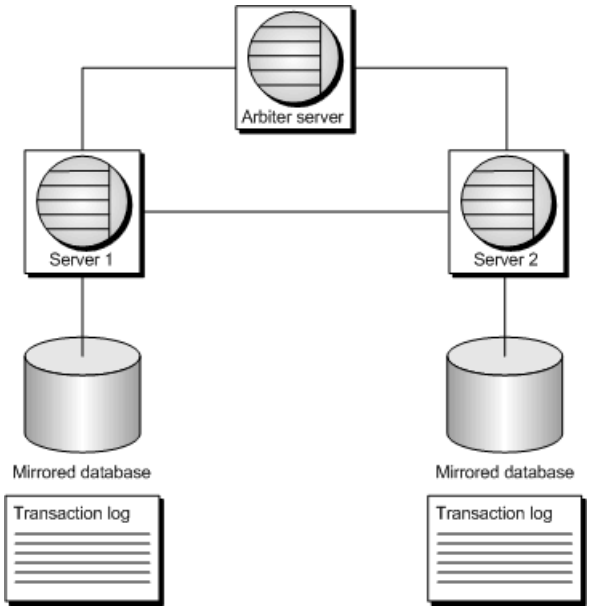
Related Information

User-initiated Role Switches (Failovers) [page 1780]
Moving a Partner Server [page 1786]
Connecting to the Utility Database (Connect Window) [page 311]
ALTER DATABASE Statement

1.10.1.7.4 Troubleshooting: When a Server Becomes Unavailable in a Database Mirroring System

Several documented scenarios can help you understand what happens when a server becomes unavailable in a database mirroring system.

These scenarios use the following database mirroring configuration, which consists of Server 1, Server 2, and an arbiter server running in synchronous mode:



At any time, you can use the MirrorState, PartnerState, and ArbiterState database properties to determine the status of the database servers in the mirroring system

Communications between the primary, mirror, and arbiter servers are checked regularly in a mirroring system. A interruption in the communication between the three servers can be caused by one or more server becoming unavailable and/or an outage of the network between the servers.

Scenario 1: Primary Server Becomes Unavailable

1. The primary server (Server 1) becomes unavailable. All clients are disconnected.
2. The arbiter and Server 2 detect that Server1 is no longer available.
3. The arbiter and Server 2 reach quorum, and Server 2 becomes the primary server.
4. Server 2 begins accepting client connections.

In this scenario, if you are running in asynchronous or asyncfullpage mode and did not specify that autofailover should occur, then you may need to make a copy of the database and restart the server that is still operational before clients can connect again.

Scenario 2: Primary Server Becomes Unavailable And Then Restarts

1. The primary server (Server 1) becomes unavailable. All clients are disconnected.
2. The arbiter and mirror server (Server 2) detect that the primary server (Server 1) is no longer available.
3. The arbiter and Server 2 reach quorum, and Server 2 becomes the primary server.
4. Server 2 begins accepting client connections.
5. Server 1 comes back online and reconnects to Server 2 and the arbiter.
6. Server 1 requests quorum, but Server 2 is already the primary server.
7. Server 1 is the mirror server, and waits for changes from Server 2.
8. Server 2 sends changes to Server 1.

Should Server 2 become unavailable before Server 1 has received all the transactions from Server 2, Server 1 cannot reach a synchronized state. It must wait for Server 2 to become available again so it can obtain and apply the transactions it does not yet have.

Scenario 3: Mirror Server Becomes Unavailable

1. The mirror server (Server 2) becomes unavailable.
2. The arbiter and Server 1 detect that the mirror server (Server 2) is no longer available.
   Client connections are not affected. They can continue to connect to the primary server. However, if either Server 1 or the arbiter server becomes unavailable, the clients cannot connect.

Scenario 4: Mirror Server Becomes Unavailable and Then Restarts

1. The mirror server (Server 2) becomes unavailable.
2. Client connections are not affected because there is no change in availability. They can continue to connect to the primary server. However, if either Server 1 or the arbiter server becomes unavailable, then clients cannot connect.
3. Server 2 comes back online and reconnects to Server 1 and the arbiter.
4. Server 2 requests quorum, but Server 1 is already the primary server.

5. Server 2 is the mirror server, and waits for changes from Server 1.
6. Server 1 sends changes to Server 2.
   Client connections are not affected because there is no change in availability. They continue connecting to Server 1.

Scenario 5: Arbiter Becomes Unavailable

1. Server 1 (primary server) and Server 2 (mirror server) detect that the arbiter is gone.
2. Both servers remain available. Client connections are not disconnected.
   When the arbiter comes back online, Server 1 and Server 2 detect it, and begin communicating with it.
   There is no change in database availability for clients.
   If Server 1 or Server 2 becomes unavailable when there isn't an arbiter server, the other server cannot reach quorum by itself, and the database is not available.

Scenario 6: Arbiter Restarts

1. Arbiter comes back online and reconnects to Server 1 and Server 2.
   Client connections are not affected because there is no change in availability.

Related Information

Troubleshooting: How to Recover from Primary Server Failure [page 1791]

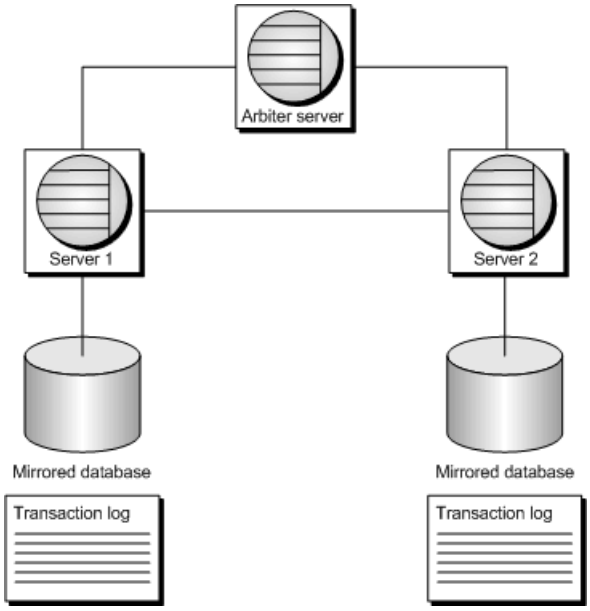
1.10.1.7.5 Troubleshooting: Database Mirroring Dropped Connections

In a database mirroring system, when connections between database servers drop, several considerations apply.

- The time required for a server to notice a dropped network connection depends on:
  1. How the connection is broken. If a network cable is unplugged from the computer, some operating systems detect this problem immediately and cause the connection to be dropped. If the problem was caused by a failure in a network switch, the problem is less likely to be detected by the server immediately.
  2. The liveness timeout setting specified in the mirror connection string.
- For the primary server to maintain ownership of a mirrored database, it must be connected to at least one of the other servers (mirror or arbiter). When connections to both of the other servers are lost, the database is restarted and the primary server waits until it can re-establish at least one of those connections to verify its status.

- The connections between servers are re-established automatically when the problem which caused the connection to be dropped is corrected.
- Until a server detects a dropped connection, it maintains the role it was previously assigned. This behavior can result in two servers attempting to act as the primary server for a brief period if the server that was previously the mirror server detects a dropped connection to the primary server before the primary server detects that its connection to the mirror server was dropped. The first update made on the original primary server blocks when the server attempts to send its changes to the other server, which limits the potential for lost transactions should this situation occur.
  When the original primary detects a dropped connection from the original mirror server and checks the arbiter's state, it sees that a failover has occurred and restarts the database.
- If a server acting as mirror must restart its database, access to the database on that server is not allowed until a connection to the primary server is re-established.
- Network failures may affect both client connections and mirror server connections, depending on the topology of the network.

The following scenarios help you understand what happens when connections are dropped in a mirroring system. The scenarios use the following database mirroring configuration, which consists of Server 1, Server 2, and an arbiter server running in synchronous mode:



At any time, you can use the MirrorState, PartnerState, and ArbiterState database properties to determine the status of the database servers in the mirroring system.

Scenario 7: Connection between the primary server and the arbiter server is dropped

1. The connection between the primary server (Server 1) and the arbiter is dropped.
2. Server 1 and the arbiter detect the loss of connection between each other.

3. Both the primary and mirror servers remain available. Client connections are not disconnected. Server 1 continues to send changes to Server 2.
    - If Server 1 becomes unavailable or loses its connection to the mirror server (Server 2), then Server 2 becomes the primary server, assuming it is still connected to the arbiter.
    - If Server 1 loses its connection to Server 2 and to the arbiter, the database on Server 1 is no longer available. Server 1 restarts the database and waits to reconnect to the arbiter and Server 2.
    - If Server 1 and Server 2 both lose their connections to the arbiter server while still connected to each other, the behavior is the same as that described in scenario 5 where the arbiter becomes unavailable.

Scenario 8: Connection between the mirror server and the arbiter server is dropped

1. The connection between the mirror server (Server 2) and the arbiter is dropped.
2. Server 2 and the arbiter detect the loss of connection between each other.
3. Both the primary and the mirror servers remain available. Client connections are not disconnected.
4. Server 1 continues to send changes to Server 2.
5. Both Server 1 and Server 2 wait for the arbiter to reconnect.
    - If Server 2 becomes unavailable or loses its connection to Server 1, client connections to Server1 are not affected because there is no change in availability. Client connections to Server 2 are lost.
    - If Server 1 and Server 2 both lose their connections to the arbiter server while still connected to each other, the behavior is the same as that described in scenario 5, where the arbiter becomes unavailable.
6. The arbiter comes back online and Server 1 and Server 2 connects to it.

Scenario 9: Connection between the primary server and the mirror server is dropped

1. The connection between the primary server (Server 1) and the mirror server (Server 2) drops.
2. Server 1 and Server 2 detect the lost connection, but potentially at different times. Both servers check the arbiter's state and then take action based on their responses from the arbiter:
    - If Server 1 detects the dropped connection first, it updates the arbiter's state to indicate that it is still the primary server and that Server 2's database may be out-of-date. When Server 2 checks the arbiter's state, it learns that Server 1 is still the primary, and Server 2 restarts its database. Server 2 waits to re-establish a connection to Server 1 before allowing access to the database.
    - If Server 2 detects the dropped connection first, it updates the arbiter's state to indicate that a failover has occurred and Server 2 becomes the primary server. When Server 1 checks the arbiter's state, it learns that a failover occurred and that Server 2 is now the primary server. Server 1 restarts its database and waits to re-establish a connection to Server 2 before allowing access to its database.

Scenario 10: Connection from the primary server to the mirror and arbiter servers is dropped

1. The connection between the primary server (Server 1) and the mirror server (Server 2) drops and the connection between the primary server (Server1) and the arbiter drops.
2. Server 1 detects that the arbiter and Server 2 are no longer available. Server 1 restarts its database and waits to reconnect to the other servers. All clients are disconnected.
3. The arbiter and Server 2 detect that Server 1 is no longer available.
4. The arbiter and Server 2 reach quorum, and Server 2 becomes the primary server.
   For the arbiter server and Server 2, the behavior is the same as that described in scenario 1, where the primary server becomes unavailable.
5. Server 2 begins accepting client connections.
6. Server 1 comes back online and reconnects to Server 2 and to the arbiter.
7. Server 1 is the mirror server and waits for changes from Server 2.
8. Server 2 sends changes to Server 1.

Scenario 11: Connection from the mirror server to the primary and arbiter servers is dropped

1. The connection between the mirror server (Server 2) and the primary server (Server 1) and the connection between the mirror server (Server 2) and the arbiter drop.
2. Server 2 detects that the arbiter and Server 1 are no longer available. Server 2 restarts the database and waits to reconnect to the primary server and the arbiter server. All client connections to Server 2 are disconnected.
   Client connections to the primary server are not affected. Clients can continue to connect to the primary server.
3. The arbiter and Server 1 detect that Server 2 is no longer available.
   For the arbiter server and Server 1, the behavior is the same as that described in scenario 3, where the mirror server becomes unavailable.
4. Server 1 updates the arbiter's state to indicate that Server 2 is no longer up-to-date.
5. Server 2 comes back online and reconnects to Server 1 and the arbiter.
6. Server 2 resumes the role of mirror server and waits for changes from Server 1.
7. Server 1 sends changes to Server 2.

Scenario 12: Connections from the arbiter server to the primary and mirror servers are dropped

1. The connection between the arbiter and the primary server and the connection between the arbiter and the mirror server (Server 2) drop.
2. When the connection to the arbiter is lost, the behavior is the same as that described in scenario 4, where the arbiter becomes unavailable.

3. When the connections are restored, the behavior is the same as that described in scenario 5, where the arbiter restarts.

Scenario 13: Connection between all servers are dropped

1. The connections between all of the servers drop.
2. The primary server (Server 1), mirror server (Server 2), and arbiter detect the loss of connection to the other servers.
3. Server 1 disconnects all clients and restarts the database.
4. Server 2 disconnects all clients and restarts the database.
5. Server 1 and Server 2 wait to reconnect to the other servers to determine their roles.
6. When connections are re-established, Server 1 and Server 2 determine their roles based on their state and the state of the other servers.

1.10.1.8 Tutorial: Creating a Database Mirroring System

Create a database mirroring system and respond to a failover.

Prerequisites

You must have the following system privileges:

- MANAGE ANY MIRROR SERVER
- BACKUP DATABASE
- CREATE TABLE
- SERVER OPERATOR

Context

In this tutorial, all the database servers are running on the same computer. However, each database server must be installed on a separate computer in a production environment.

If this tutorial is used with database servers running on different computers, references to localhost in the connection strings must be changed to the actual computer names.

1. Lesson 1: Creating a Database Mirroring System [page 1801]
   Create a mirroring system.
2. Lesson 2: Testing That the Database Mirroring System Is Running Properly [page 1804]
   Test that the database mirroring system is running properly by reviewing database and database server properties.

3. Lesson 3: Switching Roles in a Database Mirroring System [page 1805]
   Force the servers to switch roles without stopping the mirroring system.
4. Lesson 4: Stopping Your Database Mirroring System [page 1806]
   Stop the mirroring system by stopping the mirror server, the primary server, and the arbiter server.
5. Lesson 5: (Optional) Restarting the Mirroring System [page 1807]
   Start the mirroring system by starting the partner and arbiter servers.

1.10.1.8.1 Lesson 1: Creating a Database Mirroring System

Create a mirroring system.

Prerequisites

You must have the roles and privileges listed at the beginning of this tutorial.

Procedure

1. Create the following directories: `c:\server1`, `c:\server2`, and `c:\arbiter`.
2. Create a database named `mirror_demo.db` that contains data from the sample database and has a transaction log. You cannot start a database that doesn't have a transaction log in mirroring mode. Run the following command:

   ```
   newdemo.bat c:\server1\mirror_demo.db
   ```
3. Start the first database server. Run the following command:

   ```
   dbsrv17 -n mirror_server1 -x "tcpip(PORT=6871)" -su passwd "c:
   \server1\mirror_demo.db" -xp on
   ```

   -n

   Names the database server.

   -su

   Specifies the password for the utility database. It is recommended that you include the -su option to specify the password for the utility database, so that you can connect to the utility database to shut down the database server, if necessary.

   -x

   Specifies the port on which the database server runs.

   -xp on

   Indicates that the database server is available to participate in a database mirroring system.
4. Connect to the database from Interactive SQL. Run the following command:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_server1"
   ```

5. Define the partner servers and arbiter server for the database by using the CREATE MIRROR SERVER statement.

   Execute the following statements to define mirror_server1 and mirror_server2 as partner servers in the database mirroring system:

   ```
   CREATE MIRROR SERVER mirror_server1
   AS PARTNER
   connection_string='SERVER=mirror_server1;host=localhost:6871'
   state_file='c:\\server1\\server1.state';
   CREATE MIRROR SERVER mirror_server2
   AS PARTNER
   connection_string='SERVER=mirror_server2;host=localhost:6872'
   state_file='c:\\server2\\server2.state';
   ```

   Execute the following statements to define

   - mirror_demo_primary as the alternate server name for mirror_server1. mirror_demo_primary is the name that clients use to connect to the database server that is acting as the primary server.
   - mirror_demo_mirror as the alternate server name for mirror_server2. mirror_demo_mirror is the name that clients use to connect to the database server that acts as the mirror server.

   The roles of primary and mirror are necessary for configuring the database servers in the system: the names that you give these servers are used as alternate server names when clients connect to the database servers. Either partner server can act as the primary or mirror server.

   ```
   CREATE MIRROR SERVER mirror_demo_primary
   AS PRIMARY
   connection_string='SERVER=mirror_demo_primary;HOST=localhost:6871,localhost:
   6872';
   CREATE MIRROR SERVER mirror_demo_mirror
   AS MIRROR
   connection_string='SERVER=mirror_demo_mirror;HOST=localhost:6871,localhost:
   6872';
   ```

   Execute following SQL statement to define the arbiter server for the database mirroring system:

   ```
   CREATE MIRROR SERVER demo_arbiter
   AS ARBITER
   connection_string ='SERVER=demo_arbiter;HOST=localhost:6870';
   ```

6. Set the authentication string for the database. Execute the following statement:

   ```
   SET MIRROR OPTION authentication_string='abc';
   ```

7. Use the Backup utility (dbbackup) to copy the database file and transaction log in c:\server1 to c:\server2. Run the following command:

   ```
   dbbackup -c "UID=DBA;PWD=sql;SERVER=mirror_server1;DBN=mirror_demo" c:\server2
   ```

8. Start the second partner server. Run the following command:

   ```
   dbsrv17 -n mirror_server2 -x "tcpip(PORT=6872;DOBROAD=no)" -su passwd "c:
   \server2\mirror_demo.db" -xp on
   ```

9. Start the arbiter server. Run the following command:

   ```
   dbsrv17 -n demo_arbiter -su passwd -x "tcpip(PORT=6870;DOBROAD=no)" -xf "c:
   \arbiter\arbiter.state" -xa "AUTH=abc;DBN=mirror_demo"
   ```

-xf

Specifies the location of the state information file for the arbiter.

-xa

Specifies the names of the database(s) being mirrored and the authentication string (in this case abc) for the arbiter server. This authentication string must be used amongst all the servers (arbiter, primary, and mirror) in a database mirroring system.

10. Start Interactive SQL and connect to the primary server using the alternate server name for the primary server. Run following command:

```
dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_demo_primary;HOST=localhost:
6871,localhost:6872"
```

11. Add data to the database. Execute the following statements:

```
CREATE TABLE test (col1 INTEGER, col2 CHAR(32));
INSERT INTO test VALUES(1, 'Hello from server1');
COMMIT;
```

12. (Optional) To determine the name of database server that you are connected to you, use the ServerName database server property. Execute the following statement:

```
SELECT PROPERTY( 'ServerName' );
```

13. Close all Interactive SQL windows.

Results

The database mirroring system is running.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Creating a Database Mirroring System [page 1800]

Next task: Lesson 2: Testing That the Database Mirroring System Is Running Properly [page 1804]

Related Information

Database Mirroring [page 1757]
Stopping a Database Server in a Mirroring System (dbstop Utility) [page 1781]
Troubleshooting: The Primary Server Cannot be Restarted [page 1792]
Setting up a Database Mirroring System [page 1771]

Tutorial: Creating a Database Mirroring System with Multiple Databases That Share an Arbiter Server [page 1808]

1.10.1.8.2 Lesson 2: Testing That the Database Mirroring System Is Running Properly

Test that the database mirroring system is running properly by reviewing database and database server properties.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Procedure

Run the following command, to view:

- The mirror role of the database serve
- The name of the database server you are connected to
- The states of the partner, mirror and arbiter servers

```
dbping -c "UID=DBA;PWD=sql;SERVER=mirror_demo_primary" -ps ServerName -pd
MirrorRole,MirrorState,PartnerState,ArbiterState
```

Results

Information about the database properties is returned.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Creating a Database Mirroring System [page 1800]

Previous task: Lesson 1: Creating a Database Mirroring System [page 1801]

Next task: Lesson 3: Switching Roles in a Database Mirroring System [page 1805]

Related Information

ServerName (Server) Connection Parameter [page 110]
Ping Utility (dbping) [page 1195]

1.10.1.8.3 Lesson 3: Switching Roles in a Database Mirroring System

Force the servers to switch roles without stopping the mirroring system.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Procedure

1. Connect to the primary server. Run the following command:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_demo_primary;HOST=localhost:
   6871,localhost:6872"
   ```

2. Use the ServerName database server property to determine the name of the server that is currently the primary server.

   ```
   SELECT PROPERTY( 'ServerName' );
   ```

   The name of the primary server appears (mirror_server1).

   The ServerName database server property returns the real name, not the alternate name, of the server.

3. Initiate the failover by executing the ALTER DATABASE SET PARTNER FAILOVER statement

   ```
   ALTER DATABASE SET PARTNER FAILOVER;
   ```

   Connections to the primary server close. The database server messages window for server2 displays a message indicating that it is the new primary server:

4. Connect to the current primary server:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_demo_primary;HOST=localhost:
   6871,localhost:6872"
   ```

5. Execute the following statement to verify that the server that was previously acting as the mirror server (mirror_server2) is now acting as the primary server:

```
SELECT PROPERTY ( 'ServerName' );
```

6. Execute the following statement to verify that all transactions were mirrored to the mirror database:

```
SELECT * FROM test;
```

7. Disconnect from Interactive SQL.

Results

The partner servers switch roles.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Creating a Database Mirroring System [page 1800]

Previous task: Lesson 2: Testing That the Database Mirroring System Is Running Properly [page 1804]

Next task: Lesson 4: Stopping Your Database Mirroring System [page 1806]

Related Information

Troubleshooting: Database Mirroring Dropped Connections [page 1796]

1.10.1.8.4 Lesson 4: Stopping Your Database Mirroring System

Stop the mirroring system by stopping the mirror server, the primary server, and the arbiter server.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Procedure

1. Stop the database mirroring system by stopping the mirror server, the primary server, and finally the arbiter. Run the following commands in order:

```
dbstop -y -c "UID=DBA;PWD=passwd;SERVER=mirror_server2;DBN=utility_db"
```

```
dbstop -y -c "UID=DBA;PWD=passwd;SERVER=mirror_server1;DBN=utility_db"
```

```
dbstop -y -c "UID=DBA;PWD=passwd;SERVER=demo_arbiter;DBN=utility_db"
```

2. (Optional) Delete the `c:\server1`, `c:\server2`, and `c:\arbiter` directories.

Results

The mirroring system is stopped.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Creating a Database Mirroring System [page 1800]

Previous task: Lesson 3: Switching Roles in a Database Mirroring System [page 1805]

Next task: Lesson 5: (Optional) Restarting the Mirroring System [page 1807]

1.10.1.8.5 Lesson 5: (Optional) Restarting the Mirroring System

Start the mirroring system by starting the partner and arbiter servers.

Prerequisites

All database servers in the mirroring system must be stopped.

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Procedure

1. Run the following command to restart the first partner:

```
dbsrv17 -n mirror_server1 -x "tcpip(PORT=6871;DOBROAD=no)" -su passwd "c:
\server1\mirror_demo.db" -xp on
```

2. Run the following command to restart the second partner:

```
dbsrv17 -n mirror_server2 -x "tcpip(PORT=6872;DOBROAD=no)" -su passwd "c:
\server2\mirror_demo.db" -xp on
```

3. Run the following command to restart the arbiter:

```
dbsrv17 -n demo_arbiter -su passwd -x "tcpip(PORT=6870;DOBROAD=no)" -xf "c:
\arbiter\arbiter.state" -xa "AUTH=abc;DBN=mirror_demo"
```

Results

The mirroring system is restarted.

Task overview: Tutorial: Creating a Database Mirroring System [page 1800]

Previous task: Lesson 4: Stopping Your Database Mirroring System [page 1806]

Related Information

Quorum in Database Mirroring [page 1759]

1.10.1.9 Tutorial: Creating a Database Mirroring System with Multiple Databases That Share an Arbiter Server

Create a mirroring configuration in which the primary and mirror servers each host three individual databases participating in mirroring systems.

Prerequisites

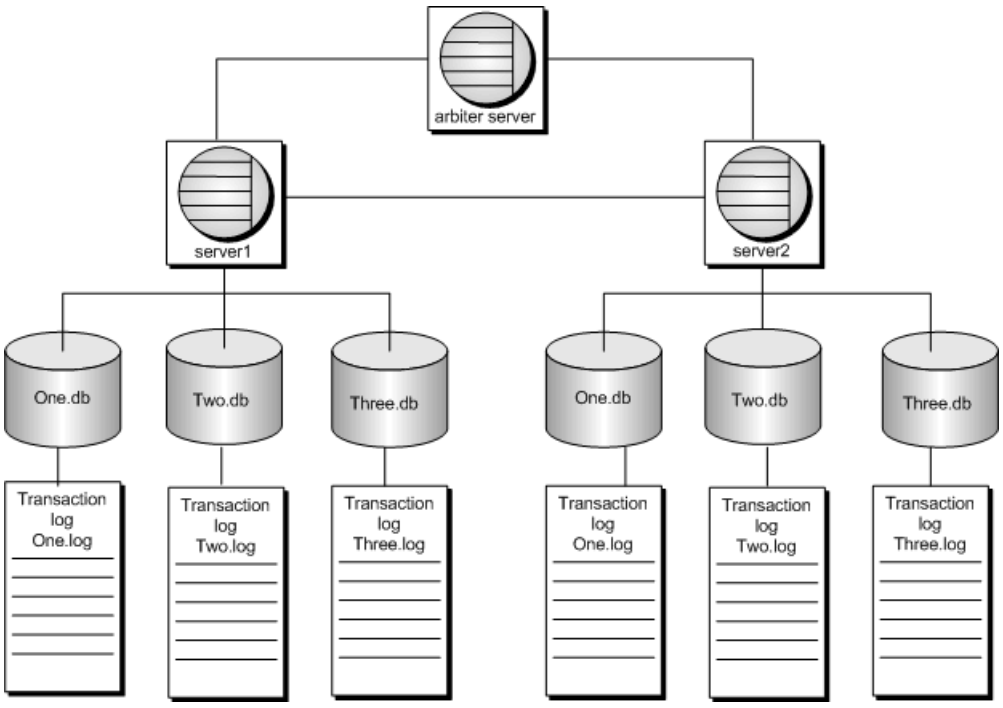
You must have the MANAGE ANY MIRROR SERVER, CREATE TABLE, and SERVER OPERATOR system privileges.

This tutorial involves backing up a copy of the database. Depending upon the backup method you choose, see the appropriate privileges for that method.

Context

All three mirroring systems communicate with the same arbiter server. Each mirroring system uses unique alternate server names that are specified using the PRIMARY and MIRROR clauses of the CREATE MIRROR SERVER statement. With this type of configuration, the primary, mirror, and arbiter servers can all run on separate computers.

> ⚠ Caution
>
> The transaction log files for one database cannot be stored in the same directory as the transaction log files for another database.



If the primary server becomes unavailable, then a role switch occurs and the mirror server takes ownership of the databases, becoming the primary server. The client must re-establish a connection to the new primary server. The alternate server name is all that must be specified to re-establish the connection to the primary server. This configuration also has the ability to protect against failure of a single database. If a database running on the primary server becomes unavailable, then a role switch occurs and the mirror server takes ownership of the failed database. The mirror server becomes the primary server for only this database. The client must re-establish a connection to the primary server for this database using the alternate server name.

There is a sample in *%SQLANYSAMP17%*`\SQLAnywhere\DBMirror` that uses database mirroring with a scale-out system. The sample can be run on one computer, or on multiple computers.

> **i Note**
>
> In this tutorial, all the database servers are running on the same computer. However, each database server must be installed on a separate computer in a production environment.
>
> If this tutorial is used with database servers running on different computers, references to localhost in the connection strings must be changed to the actual computer names.

1. Lesson 1: Creating a Database Mirroring System with Three Databases and One Arbiter Server [page 1810]
   Create a database mirroring system with three mirrored databases that use the same arbiter server.
2. Lesson 2: Using and Testing the Database Mirroring System [page 1814]
   Test the configuration of the database mirroring system by initiating a failover.

Related Information

Adding a Mirrored Database to a Running Mirroring System [page 1784]

1.10.1.9.1 Lesson 1: Creating a Database Mirroring System with Three Databases and One Arbiter Server

Create a database mirroring system with three mirrored databases that use the same arbiter server.

Prerequisites

You must have the roles and privileges listed at the beginning of this tutorial.

Procedure

1. Create the following directories:
   - `c:\server1`
   - `c:\server1\one`
   - `c:\server1\two`
   - `c:\server1\three`
   - `c:\server2`
   - `c:\server2\one`
   - `c:\server2\two`

- `c:\server2\three`
- `c:\arbiter`

2. Create copies of the SQL Anywhere 17 sample database (*demo.db*) using `newdemo.bat`. Run the following command:

```
newdemo.bat c:\server1\one\one.db
```

3. Run the following command:

```
newdemo.bat c:\server1\two\two.db
```

4. Run the following command from the `c:\server1` directory:

```
newdemo.bat c:\server1\three\three.db
```

5. Start the database server named server1:

```
dbsrv17 -n server1 -x tcpip(PORT=6871) -su passwd c:\server1\one\one.db -xp
on c:\server1\two\two.db -xp on c:\server1\three\three.db -xp on
```

It is recommended that you include the -su option to specify the password for the utility database, so that you can connect to the utility database to shut down the database server, if necessary.

6. Connect to database one from Interactive SQL and define the required mirroring objects:

```
dbisql -c "UID=DBA;PWD=sql;SERVER=server1;DBN=one"
```

a. Define server1 as a partner server for database one:

```
CREATE MIRROR SERVER server1
AS PARTNER
connection_string='SERVER=server1;host=localhost:6871'
state_file='c:\\server1\\server1state.txt';
```

b. Define the database server primary_one as the logical primary server for database one:

```
CREATE MIRROR SERVER primary_one
AS PRIMARY
connection_string='SERVER=primary_one;host=localhost:6871,localhost:6872';
```

c. Define server2 as a partner server for database one:

```
CREATE MIRROR SERVER server2
AS PARTNER
connection_string='SERVER=server2;host=localhost:6872'
state_file='c:\\server2\\server2state.txt';
```

d. Define the database server mirror_one as the logical mirror server for database one:

```
CREATE MIRROR SERVER mirror_one
AS MIRROR
connection_string='SERVER=mirror_one;host=localhost:6871,localhost:6872';
```

e. Define the arbiter server:

```
CREATE MIRROR SERVER arbiter
AS ARBITER
connection_string='SERVER=arbiter;HOST=localhost:6870';
```

f. Set the mirroring options for the database mirroring system:

```
SET MIRROR OPTION authentication_string='abc';
```

g. Disconnect from Interactive SQL.

h. Make a backup copy of the database in the `c:\server2\one` directory:

```
dbbackup -c "UID=DBA;PWD=sql;SERVER=server1;DBN=one" c:\server2\one
```

7. Connect to database two from Interactive SQL and define the required mirroring objects:

```
dbisql -c "UID=DBA;PWD=sql;SERVER=server1;DBN=two"
```

a. Define server1 as a partner server for database two:

```
CREATE MIRROR SERVER server1
AS PARTNER
connection_string='SERVER=server1;host=localhost:6871'
state_file='c:\\server1\\server1state.txt';
```

b. Define the database server primary_two as the logical primary server for database two:

```
CREATE MIRROR SERVER primary_two
AS PRIMARY
connection_string='SERVER=primary_two;host=localhost:6871,localhost:6872';
```

c. Define server2 as a partner server for database two:

```
CREATE MIRROR SERVER server2
AS PARTNER
connection_string='SERVER=server2;host=localhost:6872'
state_file='c:\\server2\\server2state.txt';
```

d. Define the database server mirror_two as the logical mirror server for database two:

```
CREATE MIRROR SERVER mirror_two
AS MIRROR
connection_string='SERVER=mirror_two;host=localhost:6871,localhost:6872';
```

e. Define the arbiter server:

```
CREATE MIRROR SERVER arbiter
AS ARBITER
connection_string='SERVER=arbiter;HOST=localhost:6870';
```

f. Set the mirroring options for the database mirroring system:

```
SET MIRROR OPTION authentication_string='def';
```

g. Disconnect from Interactive SQL.

h. Make a backup copy of the database in the `c:\server2\two` directory:

```
dbbackup -c "UID=DBA;PWD=sql;SERVER=server1;DBN=two" c:\server2\two
```

8. Connect to database three from Interactive SQL and define the required mirroring objects:

```
dbisql -c "UID=DBA;PWD=sql;SERVER=server1;DBN=three"
```

a. Define server1 as a partner server for database three:

```
CREATE MIRROR SERVER server1
```

```
AS PARTNER
connection_string='SERVER=server1;host=localhost:6871'
state_file='c:\\server1\\server1state.txt';
```

b. Define the database server primary_three as the logical primary server for database three:

```
CREATE MIRROR SERVER primary_three
AS PRIMARY
connection_string='SERVER=primary_three;host=localhost:6871,localhost:
6872';
```

c. Define server2 as a partner server for database three:

```
CREATE MIRROR SERVER server2
AS PARTNER
connection_string='SERVER=server2;host=localhost:6872'
state_file='c:\\server2\\server2state.txt';
```

d. Define the database server mirror_three as the logical mirror server for database three:

```
CREATE MIRROR SERVER mirror_three
AS MIRROR
connection_string='SERVER=mirror_three;host=localhost:6871,localhost:6872';
```

e. Define the arbiter server:

```
CREATE MIRROR SERVER arbiter
AS ARBITER
connection_string='SERVER=arbiter;HOST=localhost:6870';
```

f. Set the mirroring options for the database mirroring system:

```
SET MIRROR OPTION authentication_string='ghi';
```

g. Disconnect from Interactive SQL.

h. Make a backup copy of the database in the `c:\server2\three` directory:

```
dbbackup -c "UID=DBA;PWD=sql;SERVER=server1;DBN=three" c:\server2\three
```

9. Start the database server named server2:

```
dbsrv17 -n server2 -x tcpip(PORT=6872) -su passwd c:\server2\one\one.db -xp
on c:\server2\two\two.db -xp on c:\server2\three\three.db -xp on
```

10. Start the arbiter server.

```
dbsrv17 -n arbiter -su passwd -x tcpip(port=6870) -xf c:\arbiter
\arbiterstate.txt -xa "AUTH=abc,def,ghi;DBN=one,two,three"
```

Results

After starting server2, the server1 database server messages window shows that server1 is the primary server in the mirroring system for databases one, two, and three. The messages window also indicates that the mirror databases for one, two, and three (partners) are connected to server1.

The arbiter database server messages window shows that both server1 and server2 are connected.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Creating a Database Mirroring System with Multiple Databases That Share an Arbiter Server [page 1808]

Next task: Lesson 2: Using and Testing the Database Mirroring System [page 1814]

1.10.1.9.2 Lesson 2: Using and Testing the Database Mirroring System

Test the configuration of the database mirroring system by initiating a failover.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Procedure

1. Run the following command to start Interactive SQL and connect to database one on the primary server:

   ```
   dbisql -c "UID=DBA;PWD=sql;Server=primary_one;LINKS=TCPIP"
   ```

2. Add sample data to the database by executing the following statements:

   ```
   CREATE TABLE test (col1 INTEGER, col2 CHAR(32));
   INSERT INTO test VALUES(1, 'Hello from server1');
   COMMIT;
   ```

3. Determine which database server you are connected to by executing the following statement:

   ```
   SELECT PROPERTY( 'ServerName' );
   ```

   The name of the primary server appears.

4. Disconnect from Interactive SQL.
5. Initiate failover. Run the following command:

   ```
   dbstop -y -c "UID=DBA;PWD=sql;Server=server1"
   ```

If a warning message appears indicating that the database server still has one connection, click *Yes* to shut it down. The database server messages window for server2 displays a message indicating that it is the new primary server.

6. Restart Interactive SQL by running the following command:

```
dbisql -c "UID=DBA;PWD=sql;Server=primary_one;LINKS=tcpip"
```

7. Execute the following statement to see that you are now connected to server2:

```
SELECT PROPERTY ( 'ServerName' );
```

8. Execute the following statement to verify that all transactions were copied to the mirror database:

```
SELECT * FROM test;
```

9. Disconnect from Interactive SQL, and shut down the arbiter and server2 database servers.

Results

The database mirroring system has failed over successfully, causing server2 to become the primary server. The database mirroring system is then shut down.

Next Steps

(optional) Delete the `c:\server1`, `c:\server2`, and `c:\arbiter` directories.

Task overview: Tutorial: Creating a Database Mirroring System with Multiple Databases That Share an Arbiter Server [page 1808]

Previous task: Lesson 1: Creating a Database Mirroring System with Three Databases and One Arbiter Server [page 1810]

1.10.1.10 Tutorial: Moving the Arbiter Server

Move an arbiter server without stopping the mirroring system by creating new server and changing the arbiter server definitions to use the new arbiter server.

Prerequisites

This tutorial relies on the database mirroring system described in the tutorial on creating a database mirroring system.

You must have the MANAGE ANY MIRROR SERVER and SERVER OPERATOR system privileges.

Context

In this tutorial, all the database servers are running on the same computer. However, each database server must be installed on a separate computer in a production environment.

If this tutorial is used with database servers running on different computers, references to localhost in the connection strings must be changed to the actual computer names.

Procedure

1. Start the server to become the arbiter with the -su, -xa, and -xf options. For example:

```
mkdir arbiter2
```

```
dbsrv17 -n demo_arbiter2 -x "tcpip(port=6873)" -xf c:\arbiter2\arbiter2.state
-xa "AUTH=abc;DBN=mirror_demo" -su passwd
```

Option	Action
-su	Specifies the password for the utility database.
-xa	Specifies the database name and authentication string of the arbiter server.
-xf	Specifies the location of the state information file for the mirroring system.

2. Connect to the primary server and alter the arbiter server definition for the mirroring system.
   a. Run the following command to connect to the primary server.

```
dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_demo_primary;HOST=localhost:
6871,localhost:6872"
```

   b. Execute the following command to change the arbiter server definition to that of the new server.

```
ALTER MIRROR SERVER demo_arbiter
AS ARBITER
connection_string='SERVER=demo_arbiter2;HOST=localhost:6873';
```

   The primary and mirror servers disconnect from the arbiter server and connect to the new arbiter server.
3. Wait a few seconds, and then stop the old arbiter server.

```
dbstop -y -c "UID=DBA;PWD=sql;SERVER=demo_arbiter;HOST=localhost:
6870;DBN=utility_db"
```

4. Ping the ArbiterState database property of the mirroring system to ensure that the new arbiter server is connected to the mirroring system. Run the following command:

```
dbping -c "UID=DBA;PWD=sql;SERVER=mirror_demo_primary" -pd ArbiterState
```

The value for the ArbiterState property is connected:

```
SQL Anywhere Server Ping Utility Version 17.0.11.1293
Type       Property                Value
---------  ----------------        ------------------------------
Database   ArbiterState             connected
```

Results

The arbiter server is moved to the new server.

Related Information

Tutorial: Creating a Database Mirroring System [page 1800]

1.10.1.11 Tutorial: Moving a Partner Server

Move a server in a database mirroring system to a different server without stopping the system, delete the mirror definitions of the current mirror server, create a new server, and add the new server to the mirroring system.

Prerequisites

This tutorial relies on the database mirroring system described in the tutorial on creating a database mirroring system.

You must have the MANAGE ANY MIRROR SERVER privilege.

This task involves backing up a copy of the database. Depending upon the backup method you choose, see the appropriate privileges for that method.

Context

In this tutorial, all the database servers are running on the same computer. However, each database server must be installed on a separate computer in a production environment, and references to localhost in the connection strings must be changed to the actual computer names.

Procedure

1. Connect to the partner server that you want to move, mirror_server2, and ensure that has the mirror role. You can only move a partner with the mirror role. If the server you want to connect to is the primary server, you must initiate a failover so that the mirror and primary servers switch roles. Run the following command:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_server2;HOST=localhost:6872" SELECT
   DB_PROPERTY( 'MirrorRole' )
   ```

   The mirroring role for the current server is returned.

   - If `Mirror` is returned, then mirror_server2 is currently the mirror.
   - If `Primary` is returned, then mirror_server2 is currently the primary and you must initiate a failover to make mirror_server2 the mirror. Run the following command to initiate a failover:

     ```
     dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_server2;HOST=localhost:
     6871,localhost:6872" ALTER DATABASE SET PARTNER FAILOVER
     ```

     The roles of the two partner servers switch.
2. Create a new directory for the new partner server. Run the following command:

   ```
   mkdir c:\server3
   ```
3. Connect to the primary server. Run the following command:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_demo_primary;HOST=localhost:
   6871,localhost:6872"
   ```
4. Drop the partner definition of the server being moved by executing a DROP MIRROR SERVER statement. Execute the following statement:

   ```
   DROP MIRROR SERVER mirror_server2;
   ```

   The mirror database stops. If the mirror database is the only database running on the server, then the server also stops.
5. Create a new partner definition for the server to become the new partner server. Execute the following statements:

   ```
   CREATE MIRROR SERVER mirror_server3 AS PARTNER
   connection_string='SERVER=demo_server3;HOST=localhost:6874'
   state_file='c:\\server3\\server3.state';
   ```
6. Update the primary and mirror definitions. Execute the following statements:

   ```
   ALTER MIRROR SERVER mirror_demo_primary AS PRIMARY
   connection_string='SERVER=mirror_demo_primary;HOST=localhost:6871,localhost:
   6874';
   ALTER MIRROR SERVER mirror_demo_mirror AS MIRROR
    connection_string='SERVER=mirror_demo_mirror;HOST=localhost:6871,localhost:
   6874';
   ```
7. Make copies of the mirrored database file and transaction log from the primary server, and add them to `c:\server3` by running the following command:

   ```
   dbbackup -c "UID=DBA;PWD=sql;SERVER=mirror_server1;DBN=mirror_demo"  c:
   \server3
   ```

8. Start the new partner server with the *-xp on* option so it joins the mirroring system. Run the following command:

```
dbsrv17 -n mirror_server3 -x "tcpip(PORT=6874)" -su passwd "c:
\server3\mirror_demo.db" -xp on
```

9. Connect to the new partner server and verify that it is the mirror server. Run the following command:

```
dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_server3;HOST=localhost:6874" "SELECT
DB_PROPERTY( 'MirrorRole' )"
```

   This command returns `Mirror`.

Results

The mirror system is running with the new partner server.

Next Steps

Ensure that clients connecting to the primary server or mirror server have their connection strings updated to specify the addresses of both partners in the Host connection parameter:

The connection string to the database on the primary server:

```
'UID=DBA;PWD=sql;SERVER=mirror_demo_primary;HOST=localhost:6871,localhost:6873';
```

The connection string to the database on the mirror server:

```
'UID=DBA;PWD=sql;SERVER=mirror_demo_mirror;HOST=localhost:6871,localhost:6873';
```

Related Information

Tutorial: Creating a Database Mirroring System [page 1800]

1.10.2 SQL Anywhere Veritas Cluster Server Agents

A **cluster** is a group of computers, called **nodes**, that work together to run a set of applications.

> **i** Note
>
> The SQL Anywhere Veritas Cluster Server agents require a separate license.

Clients connecting to applications running on a cluster treat the cluster as a single system. If a node fails, other nodes in the cluster can automatically take over the services provided by the failed node. Clients may see a

slight disruption in availability (the time it takes to resume the services on the remaining nodes), but are otherwise unaware that the node has failed.

When you use clustering, any uncommitted transactions are lost when a database or database server fails over to another node in the cluster, and clients must reconnect to the database after failover occurs.

A variety of cluster environments where the cluster software can make any application into a generic resource subject to automatic failover are supported so that high availability can be provided. However, only the database server process can be failed over, and the monitoring and control processes are limited.

Most cluster software provides an API for creating custom resources tailored to a specific application. Two custom failover resources for Veritas Cluster Server are included: SAServer and SADatabase. The SAServer agent is responsible for database server failover, while the SADatabase agent is responsible for the failover of a specific database file. You can use one or both agents, depending on your application.

Your systems must be set up as follows to use the SQL Anywhere Veritas Cluster Server agents:

- You must use Veritas Cluster Server 4.1 or later.
- SQL Anywhere must be installed identically on each system node within the cluster.
- Database files must be stored on a shared storage device that is accessible to all systems within the cluster.
- The utility database password must be the same for all systems within the cluster.
  The SADatabase agent uses the utility database to start and stop specific database files. All systems participating in the cluster must have the same utility database password. Set the utility database password by specifying the -su database server option when starting the database server.
  On UNIX and Linux, the VCS agent is installed in *$SQLANY17*`/vcsagent/saserver`.

There are three ways to configure and add a new agent to Veritas Cluster Server:

1. Using the Cluster Manager.
2. Using command line utilities.
3. Using a text editor and editing the `main.cf` configuration files.

These instructions use the Cluster Manager.

For information about the available utilities, see *Veritas Cluster Server Administration Guide*.

To configure `main.cf` manually using a text editor, you must stop all Veritas Cluster Server services before editing the `main.cf` file. Otherwise, the changes do not take effect.

In this section:

Related Information

Separately Licensed Components
Setting Up SQL Anywhere as a Cluster Database Service

-su Database Server Option [page 502]

1.10.2.1 SAServer Agent Configuration

There are several ways to configure the SAServer agent.

In this section:

1.10.2.1.1 Setting up the SAServer Agent

Set up the SAServer agent so that it can control the failover of a database server to another node in the cluster.

Prerequisites

Your systems must be set up as follows to use the SQL Anywhere Veritas Cluster Server agents:

- You must use Veritas Cluster Server 4.1 or later.
- SQL Anywhere must be installed identically on each system node within the cluster.
- Database files must be stored on a shared storage device that is accessible to all systems within the cluster.
- The utility database password must be the same for all systems within the cluster.
  The SADatabase agent uses the utility database to start and stop specific database files. All systems participating in the cluster must have the same utility database password. You can set the utility database password by specifying the -su server option when starting the database server.
- On UNIX and Linux, the VCS agent is installed in *$SQLANY17*`/vcsagent/saserver`.

Context

The SAServer agent controls the failover of a database server to another node in the cluster.

Procedure

1. Shut down all database servers running on nodes in the cluster.
2. Choose a node in the cluster and create a directory named `SAServer` under the `%VCS_HOME%\bin` directory on that node. You may see other Veritas Cluster Server agents within this folder (such as NIC and IP).
3. Copy the following files from the *%SQLANY17%*`\VCSAgent\SAServer` directory to the `SAServer` directory you created in Step 2:
   - `Online.pl`
   - `Offline.pl`
   - `Monitor.pl`
   - `Clean.pl`
   - `SAServer.xml`
4. Copy the file `%VCS_HOME%\bin\VCSdefault.dll` into the `%VCS_HOME%\bin\SAServer` directory and rename it to `SAServer.dll`.
5. Copy the file *%SQLANY17%*`\VCSAgent\SAServer\SAServerTypes.cf` into the `%VCS_HOME%\conf\config` directory.
6. Repeat Steps 1-5 for all other nodes in the cluster.
7. Start the Veritas Cluster Server Manager and enter your user name and password to connect to the cluster.
8. Add the SAServer agent:
   a. Click *File* > *Import Types*.
   b. Navigate to `%VCS_HOME%\conf\config\SAServerTypes.cf`, and then click *Import*.

Results

The SAServer agent is set up.

Related Information

-su Database Server Option [page 502]

1.10.2.1.2 Setting up a Database Server for Failover Using the SAServer Agent

Set up a database server to failover to another database server using the Veritas Cluster Server.

Prerequisites

Your systems must be set up as follows to use the SQL Anywhere Veritas Cluster Server agents:

- You must use Veritas Cluster Server 4.1 or later.
- SQL Anywhere must be installed identically on each system node within the cluster.
- Database files must be stored on a shared storage device that is accessible to all systems within the cluster.
- The utility database password must be the same for all systems within the cluster.
  The SADatabase agent uses the utility database to start and stop specific database files. All systems participating in the cluster must have the same utility database password. You can set the utility database password by specifying the -su server option when starting the database server.
- On UNIX and Linux, the VCS agent is installed in *$SQLANY17*`/vcsagent/saserver`.

When stopping a network server, you must supply a connection string with a user ID that has permissions to stop the server. By default, the SERVER OPERATOR system privilege is required to stop the network server. You can use the -gk server command-line option to change the default behavior.

Procedure

1. Start the Veritas Cluster Server Manager and enter your user name and password to connect.
2. Add SAServer as a resource to a service group:
   a. Click *Edit* > *Add* > *Resource*.
   b. In the *Resource Type* list, click *SAServer*.

      On Microsoft Windows, if SAServer does not appear in the *Resource Type* list under Microsoft Windows, you may have to add the `SAServer.xml` file to the `%VCS_ROOT%\cluster manager \attrpool\Win2K\400` and restart the cluster services.
   c. In the *Resource Name* field, type a name.
   d. Add the following attribute values to the following attributes:

      cmdStart

      dbsrv17 -x tcpip `database-file-on-shared-disk` -n `server-name`

      cmdMonitor

      dbping -c "Server=`server-name`"

      cmdStop

      dbstop -c `user-id`,`password` -y

e. Click *Enabled*.

   This indicates that the resource is ready to be used.

f. Click *OK*.

3. Ensure that the resource dependencies are configured correctly. There are other resources that must be started and grouped together before SAServer can be started, such as the shared disk resources and the IP address resources.
4. Right-click the service group and click *Online* `node-name`, where `node-name` is the name of the computer in the cluster on which you want the resource to run.

Results

The service group is now online.

Related Information

-su Database Server Option [page 502]

1.10.2.1.3 Testing the SAServer Agent

Test failover in the SAServer agent using the Veritas Cluster Server.

Prerequisites

Your systems must be set up as follows to use the SQL Anywhere Veritas Cluster Server agents:

- You must use Veritas Cluster Server 4.1 or later.
- SQL Anywhere must be installed identically on each system node within the cluster.
- Database files must be stored on a shared storage device that is accessible to all systems within the cluster.
- The SADatabase agent uses the utility database to start and stop specific database files. All systems participating in the cluster must have the same utility database password. You can set the utility database password by specifying the -su server option when starting the database server.
- On Unix, the VCS agent is installed in *$SQLANY17*`/vcsagent/saserver`.

You must have the SELECT object-level privilege on the table you are selecting from, or the SELECT ANY TABLE system privilege.

Procedure

1. Connect to the database from Interactive SQL. For example:

   ```
   dbisql -c "UID=DBA;PWD=passwd;Server=VCS;HOST=HostName"
   ```

2. Execute the following query:

   ```
   SELECT * FROM Departments;
   ```

   The query should execute without errors.
3. Shut down the system running the database server.
4. Reconnect from Interactive SQL using the same connection string and executing the query again. You should be able to connect and execute the query successfully.

Results

Failover should have occurred, and all resources should have started on the alternate server.

Related Information

-su Database Server Option [page 502]

1.10.2.2 SADatabase Agent Configuration

You can configure the SADatabase agent.

In this section:

1.10.2.2.1 Setting up the SADatabase Agent

Set up the SADatabase agent to control the failover of a database to another node in the cluster.

Prerequisites

Your systems must be set up as follows to use the SQL Anywhere Veritas Cluster Server agents:

- You must use Veritas Cluster Server 4.1 or later.
- SQL Anywhere must be installed identically on each system node within the cluster.
- Database files must be stored on a shared storage device that is accessible to all systems within the cluster.
- The utility database password must be the same for all systems within the cluster.
  The SADatabase agent uses the utility database to start and stop specific database files. All systems participating in the cluster must have the same utility database password. You can set the utility database password by specifying the -su server option when starting the database server.
- On UNIX and Linux, the VCS agent is installed in *$SQLANY17*`/vcsagent/saserver`.

The `offline.pl` file requires by default the SERVER OPERATOR system privilege to stop the database. You can use the -gk server command-line option to change the default behavior.

The `online.pl` file requires by default the SERVER OPERATOR system privilege to start the database on a network server. You can use the -gk server command-line option to change the default behavior.

Procedure

1. Shut down all database servers running on nodes in the cluster.
2. Create a directory named `%VCS_HOME%\bin\SADatabase` on one of the nodes in the cluster.
3. Copy the following files from the *%SQLANY17%*`\VCSAgent\SADatabase` directory to the `%VCS_HOME%\bin\SADatabase` directory you created in Step 2:
   - `Online.pl`
   - `Offline.pl`
   - `Monitor.pl`
   - `Clean.pl`
   - `SADatabase.xml`
4. Copy the file `%VCS_HOME%\bin\VCSdefault.dll` into the `%VCS_HOME%\bin\SADatabase` directory and rename it to `SADatabase.dll`.
5. Copy the file *%SQLANY17%*`\VCSAgent\SADatabase\SADatabaseTypes.cf` into the `%VCS_HOME%\conf\config` directory.
6. Repeat steps 1-5 for all systems participating in the cluster.
7. Start the Veritas Cluster Server Manager and enter your user name and password to connect to the cluster.

8. Add the SADatabase agent:
    a. Click *File* > *Import Types*.
    b. Navigate to `%VCS_HOME%\conf\config\`, and click *Import*.

Results

The SADatabase agent is set up.

Related Information

-su Database Server Option [page 502]

1.10.2.2.2 Setting up a Database for Failover Using the SADatabase Agent

Add a database to the service group before it is configured to fail over.

Prerequisites

Your systems must be set up as follows to use the SQL Anywhere Veritas Cluster Server agents:

- You must use Veritas Cluster Server 4.1 or later.
- SQL Anywhere must be installed identically on each system node within the cluster.
- Database files must be stored on a shared storage device that is accessible to all systems within the cluster.
- The utility database password must be the same for all systems within the cluster.
  The SADatabase agent uses the utility database to start and stop specific database files. All systems participating in the cluster must have the same utility database password. You can set the utility database password by specifying the -su server option when starting the database server.
- On Unix, the VCS agent is installed in *$SQLANY17*`/vcsagent/saserver`.

Procedure

1. Add SADatabase as a resource to the service group:
    a. Click *Edit* > *Add* > *Resource*.

b. From the *Resource Type* list, click *SADatabase*.

   On Windows, if SADatabase does not appear in the *Resource Type* list, you may have to add the `SADatabase.xml` file to the `%VCS_ROOT%\cluster manager\attrpool\Win2K\400` and restart the cluster services.

c. In the *Resource Name* field, type a name.

d. Add the specified values to the following attributes by clicking the button in the *Edit* column for each attribute:

   DatabaseFile

   The location of the database file, for example, `E:\demo.db`.

   DatabaseName

   A name for the database.

   ServerName

   A name for the database server. A different server name can be supplied on each system within the cluster. The scope of the attribute should be Per System, not Global.

   UtilDBpwd

   The utility database password used for all systems within the cluster.

e. Click *Enabled*.

   This indicates that the resource is ready to be used.

f. Click *OK*.

2. Ensure that the resource dependencies are configured correctly. There are other resources that must be started/grouped together before SADatabase can be started, such as the shared disk resources and the IP address resources.

3. Right-click the service group, and click *Online* `node-name`, where `node-name` is the name of the computer in the cluster on which you want the resource to run.

Results

The service group is now online.

Related Information

-su Database Server Option [page 502]

1.10.2.2.3 Testing the SADatabase Agent

Test the SADatabase agent failover using Interactive SQL.

Prerequisites

Your systems must be set up as follows to use the SQL Anywhere Veritas Cluster Server agents:

- You must use Veritas Cluster Server 4.1 or later.
- SQL Anywhere must be installed identically on each system node within the cluster.
- Database files must be stored on a shared storage device that is accessible to all systems within the cluster.
- The utility database password must be the same for all systems within the cluster.
  The SADatabase agent uses the utility database to start and stop specific database files. All systems participating in the cluster must have the same utility database password. You can set the utility database password by specifying the -su server option when starting the database server.
- On UNIX and Linux, the VCS agent is installed in *$SQLANY17*`/vcsagent/saserver`.

You must have the SELECT object-level privilege on the table you are selecting from, or the SELECT ANY TABLE system privilege.

The required privileges to stop a database on the network server are determined by the database server -gd option. The default system privilege for stopping a database on the network server is SERVER OPERATOR.

Procedure

1. Connect to the database from Interactive SQL. For example:

   ```
   dbisql -c "UID=DBA;PWD=sql;Server=VCS;HOST=HostName"
   ```

2. Execute the following query:

   ```
   SELECT * FROM Departments;
   ```

   The query should execute without errors.

3. Suppose the database failed, and the database server running on the first system node cannot access the database file. This would create a failover of the database file to the database server started on the second system node. You can cause the database file on the first node to fail by issuing a command similar to the following command:

   ```
   dbisql -q -c "UID=DBA;PWD=sql;Server=VCS1;DBN=utility_db" STOP DATABASE DEMO
   ON VCS1 UNCONDITIONALLY;
   ```

Results

The database file on the first computer fails. There is a delay before Veritas Cluster Server recognizes that the file has failed because Veritas Cluster Server monitors the health of its resource, every 60 seconds by default (you can make this interval smaller in your resource configuration). The database file then fails over to the second computer, and that database file is started using the database server on the second computer, which may have a different name than the original database server.

For example, if the new database server is called VCS2, then clients must specify the new database server name in their connection strings:

```
"UID=DBA;PWD=sql;Server=VCS2;DBN=DEMO;HOST=HostName"
```

Next Steps

Reconnect from Interactive SQL. You should be able to connect and execute the query successfully.

Related Information

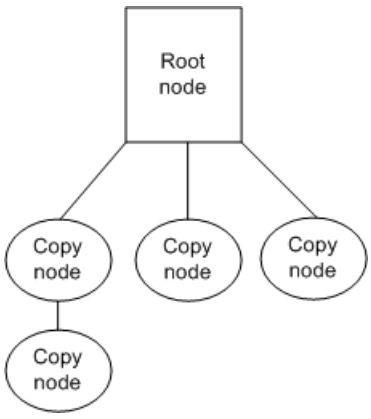
-su Database Server Option [page 502]

1.10.3 Read-only Scale-out

Read-only scale-out allows you to offload reporting or other operations that require read-only access to the database.

A scale-out tree consists of the root node and copy nodes.

- The **root node** of a scale-out system has the only copy of the database that is writable. The root node can be either a single database, or a database mirroring system. Below the root node, there can be one or many branches of copy nodes.
- A **copy node** is a database server that runs a copy of a database for read-only access, and a copy node can be the parent of other copy nodes.



Read-only scale-out requires a separate license.

In this section:

Related Information

Separately Licensed Components
Database Health and Statistics [page 1109]
Lesson 8: (Optional) Monitoring Your Read-only Scale-out System from the SQL Anywhere Monitor [page 1368]

1.10.3.1 How Scale-out Works

You can use scale-out with database mirroring to ensure the availability of your database.

The root node is the only database server that accepts both read and write requests. Once you start additional copy nodes, the root database server sends transaction log pages to the copy nodes in the tree that are defined as its children, provided that they are connected and ready to receive transaction log pages. The pages are normally sent without waiting for a response; however, the root database server occasionally requests an acknowledgement to ensure that the copy node does not receive more asynchronous requests than it can handle.

When the copy node receives pages, it writes them to disk and then sends them to its children (if it has any).

The parent detects if a child node becomes unavailable, and if this happens, the parent stops pushing transaction log pages to the child. If the child is restarted, it requests the transaction log pages that it does not have, and then the parent resumes pushing transaction log pages to the child. The child notifies its parent of changes in the status of the copy node, and the status information eventually makes its way up through the tree to the root database server.

If the root database server becomes unavailable, all of the children in the scale-out system continue running, but they no longer receive updates from the primary database server. Any connections to the copy nodes may retrieve data that is stale. When the root database server becomes available again, its children re-establish connections and resume receiving transaction log pages.

If a copy node encounters a problem, such as an incompatible or mismatched transaction log, the database is stopped. The database server running the problem database also shuts down unless there are other databases running on it.

Related Information

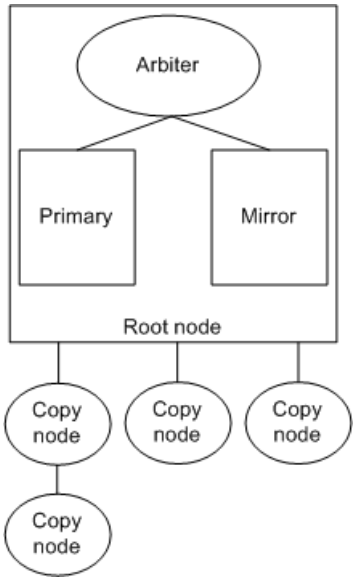
Read-only Scale-out in a Database Mirroring System [page 1832]
Database Health and Statistics [page 1109]
Lesson 8: (Optional) Monitoring Your Read-only Scale-out System from the SQL Anywhere Monitor [page 1368]

1.10.3.2 Read-only Scale-out in a Database Mirroring System

In a database mirroring system, you can access the database running on the mirror server using a read-only connection.

This functionality is useful to offload reporting or other operations that require read-only access to this database. The only difference between a mirror server and a copy node is that a copy node cannot participate as a primary or mirror server in a database mirroring system. A copy node can act as an arbiter for a database that it copies.

You can use database mirroring with read-only scale-out to ensure the availability of the root node. When scale-out is used with database mirroring, the root node of the scale-out system consists logically of the primary, mirror, and arbiter servers, instead of a single database server.



Once you start the database servers in the mirroring system, additional database servers can access the participating mirror servers to maintain read-only copies of the database. Add as many branches of copy nodes to the mirror servers as necessary.

Differences Between Copy Nodes and Primary and Mirror Servers

When copy nodes are used in a database mirroring system, they can be the child of the current primary server, the current mirror server, or another copy node. The copy node gets its transaction log pages from its parent. Unlike mirror nodes, copy nodes do not have a state information file because their state does not influence which database server has ownership of the database. A copy node can also be used as an arbiter for a database that it is copying. If a copy node is acting as arbiter it must have an arbitrary mirror server name that does not match the server name of any of the database servers in the high availability system.

Related Information

Database Mirroring [page 1757]
Web Services High Availability and Scale-out Solutions with Relay Server and the Outbound Enabler [page 1834]
Using a copy node as an arbiter [page 1776]
SYSMIRRORSERVER System View
CREATE MIRROR SERVER Statement

1.10.3.3 Web Services High Availability and Scale-out Solutions with Relay Server and the Outbound Enabler

There are several components are required to configure high availability and/or scale-out solutions for hosting web applications.

Relay Server (RS)

The Relay Server is the entry point for HTTP requests (typically from the Internet), providing virtual hosting and load balancing capability. The Relay Server accepts and routes HTTP requests that target virtual entities called a **server farm**. Each server farm consists of one or more database/web servers. The Relay Server is the HTTP/WEB access point that relays all HTTP requests to an appropriate backend server for processing.

Outbound Enabler (OE)

The Outbound Enabler provides a bridge between the Relay Server and a backend database server. There is one Outbound Enabler process for each backend server residing within a server farm. Its purpose is to ensure that its designated backend server is operational, and the Outbound Enabler establishes a communication channel with the Relay Server and forwarding requests to its backend server. Since the Outbound Enabler initiates the connection to the Relay Server, it can easily make an outbound connection from a LAN, through a firewall, to a Relay Server situated in the DMZ.

SQL Anywhere (backend) server

SQL Anywhere (backend) server is an advanced database and web server that can host web applications written entirely in SQL. Servers may be deployed to mitigate demanding environments requiring high availability and scale-out solutions. High availability provides fault tolerance capabilities to protect against interruption of service and scale-out provides load balancing to increase throughput during peak loads.

High Availability

Three servers are required to provide high-availability: an **arbiter** and two **partner** servers. One partner is assigned the role of the **primary** and the other is the **mirror**. Write operations, such as an insert to a global table, are only permitted on the primary, while read operations, such as a select, are permitted on both types of servers. The primary automatically keeps its mirror synchronized as it receives updates. The arbiter server is used when it is necessary to determine which of the other two servers can take ownership of the database.

High availability may be leveraged within a web service environment by routing all HTTP requests through a Relay Server. The Relay Server processes the URL of the HTTP request to derive the target server farm. For simplicity, consider a Relay Server configuration that routes HTTP requests to either a readFarm or a writeFarm. Although the primary backend server may process both read and write requests, you can configure it to process write requests only if there are other read-only copy nodes to cover the read functionality in case of server failure; otherwise, you can configure it to process both read and write requests. However, you may configure the mirror server to process read requests to spread the read load, or it can be configured not to process any requests to reduce hardware use on the mirror server. The client application requires no knowledge of the physical server that it accesses for read and write requests; it composes its URL to direct the request to either the readFarm or the writeFarm.

In addition to the Relay Server, each of the partnered backend servers requires two Outbound Enablers: one for each farm the backend server participates in. Two Outbound Enablers control communication to the primary

partner and two Outbound Enablers establish communications with the mirror partner. Within this configuration, at any instant in time there is only one primary server and it activates its Outbound Enabler assigned to the writeFarm, and optionally deactivates its Outbound Enabler associated with the readFarm if there are other members in the readFarm, other than the copy nodes. The mirror deactivates its Outbound Enabler associated with the writeFarm and optionally activates its Outbound Enabler associated with the readFarm.

If the primary server becomes unavailable because of hardware or software failure, the mirror server negotiates with the arbiter to assume the role of primary server, and thus become a member of the writeFarm. It should also become a member of the readFarm if the system has no other read-only copy nodes. At the time of failure, the original primary is abruptly withdrawn from both the writeFarm and the readFarm if it is a member. The original primary server may not recover until the cause of the failure has been resolved. When it recovers, it assumes the mirror role, and optionally joins the readFarm. The abrupt failure may fail requests currently being processed by the failing server. The latency of writeFarm recovery from the mirror depends on the role-switch latency of the mirror plus the status polling interval specified on the writeFarm Outbound Enabler of the mirror. This status polling interval can be set and its default is 5 seconds if not explicitly specified.

Scale-Out

A minimum of two servers are required to provide scale-out capability: a root node and a copy node. A copy node, like a mirror server, is read-only, but it differs in that it can never assume the primary role. It only services read requests. The primary is associated with the writeFarm and the copy node replaces the mirror server as the readFarm member. With one root node and a single copy node, only two Outbound Enablers are required in total, one for each server. Optionally, you may add the primary to the readFarm so that your environment is fully functional when all the copy nodes are down. The Outbound Enabler associated with the root node is configured to receive requests to the writeFarm while the copy node's Outbound Enabler is configured to accept requests to the readFarm. The root node continually updates its copy node as it receives write updates. The configuration can be expanded by adding copy nodes to the readFarm. Running 10 copy nodes to the readFarm provides load balancing and mitigates server failures incurred within the readFarm.

High-Availability Plus Scale-Out

The above configuration scenarios may be combined to provide both high-availability and scale-out. Such a configuration provides fault tolerance for writable (primary/mirror) and readable (mirror and copy) backend servers. In addition, adding more copy nodes increases the throughput potential for HTTP read-only requests. In this configuration, the primary and mirror server make up the root node, and the primary, the mirror, and the copy nodes all have one Outbound Enabler for participation in the readFarm. The primary and mirror have one additional Outbound Enabler for participation in the writeFarm. Optionally, you can remove the primary and mirror from the readFarm.

Outbound Enabler Configuration

For the purposes of this discussion, an Outbound Enabler is configured to identify the following:

- its associated server farm
- Relay Server connection parameters
- SQL Anywhere (backend server) connection parameters

In the following command, the Outbound Enabler sets up a connection to the Relay Server based on parameters specified with the -cr option. If the Outbound Enabler is active, the Relay Server forwards requests made to the readFarm to the backend server configured according to the -cs option.

```
rsoe2.exe -f readFarm -id saHost1 -cr url_suffix="/rs/server/
rs.dll";host=rs_server;port=80 -cs host=sahost1;port=8080;
```

Based on the above example, a request URL of the form `http://rs_server/rs/client/rs.dll/readFarm/service_path[?service_query]` directs the request to the Relay Server hosted on rs_server. The Relay Server strips off the domain and farm components from the URL and relays the request to an active Outbound Enabler associated with the farm, over the Relay Server-Outbound Enabler junction. The Outbound Enabler then forwards the request to its backend server.

An Outbound Enabler connects a junction to the Relay Server only if it can connect the junction to the backend server first. This behavior reduces access latency when the request from the client arrives, as well as preventing traffic from being directed to the Outbound Enabler if it cannot communicate with the backend server.

SQL Anywhere configuration

Leveraging the Relay Server Outbound Enabler (RSOE) startup example discussed in the Relay Server Outbound Enabler configuration section, an oe_read_status service may be written as follows:

```
// READ-ONLY SA-OE ping service
//  optionally includes primary as a READ node.
call sa_make_object( 'service', 'oe_read_status' );
alter service oe_read_status
    type 'raw'
    authorization off
    secure off
    user DBA
    as call sp_oe_read_status(:ro);
// if ro is 1, then only read-only servers are activated: mirror or copy
// if ro is 0, then include all servers: primary, mirror, copy...
create or replace procedure sp_oe_read_status( ro int )
begin
    declare readonly long varchar;
    declare res long varchar;
    set res = 'AVAILABLE=';
    set ro = isnull(ro, 0);
    if ro = 1 then
         select db_property('ReadOnly') into readonly;
          if readonly = 'On' then
               set res = res || 'TRUE';
          else
               set res = res || 'FALSE';
           end if;
```

```
    end if;;
    call sa_set_http_header('Content-Length', length(res) );
    select res;
end;
```

The above web service and stored procedure could be used for any server that has the role of a primary, mirror, or copy node. If the ReadOnly database property is set to On, then `AVAILABLE=TRUE` is returned. With minor modifications, a service for use within a writable server farm, for example writeFarm, would return `AVAILABLE=TRUE` only when the ReadOnly database property is set to Off. Optionally, a ping service for an Outbound Enabler may base its AVAILABILITY on other factors such as an entry in a global table. For example, a staged deployment might update a table or procedure on the primary which, when propagated to the mirror and copy nodes, activates or deactivates each server's server farm membership.

Relay Server Configuration

To configure the Relay Server:

1. Create the readFarm and writeFarm backend farms.
2. Configure the primary, mirror, and copy nodes to be members of the readFarm Relay Server backend farm.
3. Configure the primary and mirror to be members of the writeFarm Relay Server backend farm.
4. For both farms, set the active_cookie configuration option to **yes** and the active_header configuration option to **no**. If you do not require affinity and want to improve performance, set the active_cookie configuration option to no.

Client/Application Configuration

Limitations

backend servers only support high availability running a single database. The maximum latency for high availability fail-over is the maximum of:

- The polling frequency of the Outbound Enabler, which is controlled by the -d option in the Outbound Enabler syntax.
- The time it takes for the mirror to become the primary.

Related Information

The Database Server as an HTTP Web Server
Introduction to the Relay Server
The Relay Server Configuration File
Relay Server Farm Configuration Updates
The Outbound Enabler
Configuring a Backend Farm (Command Line)
Relay Server Outbound Enabler Syntax

1.10.3.4 Connecting to a Database in a Read-only Scale-out System

To set up a scale-out system, you create two definitions with the CREATE MIRROR SERVER statement for the database server that is acting as the root node.

- one definition for the partner role
  In a scale-out system, you must define one partner server. The name that is specified in the partner server definition is used in connection strings and when starting the database server.
- one definition for the primary role.
  The database server defined as the primary server is the default parent for copy nodes in the scale-out system. If there is no primary server defined, then you must specify the parent for each copy node that is added to the system.

When you use read-only scale-out, it is recommended that the application connects to the root database server, and the root database server uses information from the application's connection string, together with status and load information from the copy nodes, to determine which node the application should connect to. You can choose to have the application connect to the copy node that is least heavily loaded using the `NodeType=COPY` connection parameter. Specifying this causes the root database server to redirect the client to that node.

If an application makes and drops several such connections within a short period of time, the connection is pooled and the root database server is not asked which copy node to use. This behavior reduces the load on the root database server, but may not give expected behavior. The application can specify that its connections are not to be pooled to ensure that the root server determines which copy node to connect to on each connection.

> → Tip
>
> Check the status of the database servers in a scale-out system by monitoring the primary database using the SQL Anywhere Monitor.
>
> Monitor a scale-out system from SQL Central by connecting to the primary database and checking the status of the database servers in the *Health And Statistics* pane.

In this section:

Related Information

Read-only Scale-out in a Database Mirroring System [page 1832]
Improve Application Performance with Connection Pooling [page 256]
Database Health and Statistics [page 1109]
Lesson 8: (Optional) Monitoring Your Read-only Scale-out System from the SQL Anywhere Monitor [page 1368]

1.10.3.4.1 Connect to Copy Nodes

Connect to a copy node by connecting to the primary server and specifying the NodeType connection parameter as part of the connection string.

The NodeType connection parameter accepts the following values:

DIRECT

This is the default setting when connecting to database server.

When the NodeType connection parameter is set to DIRECT, the database server accepts the connection without performing load balancing or redirection.

PRIMARY

If the NodeType connection parameter is set to PRIMARY and you have connected to the primary server, the connection is accepted. If you have connected to a non-primary server (such as, the mirror server, or a copy node), the database server redirects the connection to the primary server.

COPY

When the NodeType connection parameter is set to COPY, the database server performs load balancing and chooses a copy node. In a read-only scale-out system, the database server examines the copy nodes in its own branch (including itself if it is not the root node) and chooses the copy node with the lightest load. If the database server does not choose itself, it redirects the client to the chosen database server.

MIRROR

If the NodeType connection parameter is set to MIRROR and you have connected to the mirror server, the connection is accepted. If you have connected to a non-mirror server, the database server redirects the connection to the mirror server.

READONLY

When the NodeType connection parameter is set to READONLY, you are connected to any read-only server, either a copy node or the mirror. If there are no copy nodes, this is equivalent to the MIRROR setting. If there are copy nodes, this is equivalent to the COPY setting.

Once the root database server determines which copy node the connection should be made to, the connection is automatically redirected (if necessary) to that node.

Related Information

Connecting to a Database in a Mirroring System [page 1762]
NodeType (NODE) Connection Parameter [page 101]

1.10.3.5 Read-only Scale-out Setup

There are several steps that must be performed to configure a scale-out system.

1. Start a database server with the *-xp on* database option. This database server is the root node for the scale-out system.

2. Add the scale-out object definitions to the root database. You must define the servers and set options for the servers.
3. Make backup copies of the root database.
4. Start the copy nodes (the backup copies of the database).

User Authentication Considerations

If you are using the NodeType connection parameter to redirect connections that use any of the following user authentication methods, then all the database servers in the scale-out tree must be configured as follows:

Integrated login

(Microsoft Windows only) All of the database servers in the scale-out tree must be a part of the same Microsoft Windows user group or be configured to allow the same Microsoft Windows users to access them.

Kerberos user authentication

All of the database servers in the scale-out tree must have Kerberos user authentication enabled. The Kerberos server principal for all the servers in the scale-out tree must have the same realm.

LDAP user authentication

All of the database servers in the scale-out tree must have LDAP user authentication enabled. To enable TLS encryption for communications with the LDAP server, each database server needs access to the trusted certificates file. Therefore, the trusted certificates file must be stored using the same file path on each computer.

Whenever authentication with the LDAP server succeeds, an attempt is made to update the user's password in the database if the stored value is different. This may succeed or fail depending on whether the database is read-only. Also, the updated password is not replicated to other nodes. So if the password has changed and the LDAP server is unavailable, failover to Standard user authentication may fail.

PAM user authentication

(UNIX and Linux only) All of the database servers in the scale-out tree must have PAM user authentication enabled. Therefore, each computer must have the appropriate PAM configuration defined.

Whenever authentication with the PAM service succeeds, an attempt is made to update the user's password in the database if the stored value is different. This may succeed or fail depending on whether the database is read-only. Also, the updated password is not replicated to other nodes. So if the password has changed and the PAM service is unavailable, failover to Standard user authentication may fail.

In this section:

Copy Node Parent Assignment [page 1844]
A copy node must have a parent.

1.10.3.5.1 Setting up the Root Node

Create the root node of a read-only scale-out system using the database that you want to have copies of.

Prerequisites

You must have the MANAGE ANY MIRROR SERVER system privilege.

Context

When you configure the root node, define the root database server as a partner server. You can also define a primary server.

Procedure

1. Start a database server running the database that you want to have read-only copies of. You must specify the *-xp on* database option when starting the database server.

   For example, the following command starts a database server named scaleout_root_demo that starts on port 6871. It is recommended that you use the -x option to specify a port number, as this port number is used later on in the CREATE MIRROR SERVER statements.

   ```
   dbsrv17 -n scaleout_root_demo -x TCPIP(port=6871) "c:\scaleoutdemo
   \scaleoutdemo.db" -xp on
   ```

2. Define the scale-out objects in the root database using the CREATE MIRROR SERVER statement.

   It is recommended that you create two definitions for the root database server: CREATE MIRROR SERVER...AS PARTNER and CREATE MIRROR SERVER...AS PRIMARY. You must specify the connection_string parameter in both definitions.

   a. The name that you give the database server in the statement that uses the AS PARTNER clause is the name that is used in the command to start the database server and in client connection strings. For example:

      ```
      CREATE MIRROR SERVER "scaleout_root_demo"
      AS PARTNER
      connection_string = 'SERVER=scaleout_root_demo;HOST=localhost:6871';
      ```

   b. The name that you give the database server in the statement that uses the AS PRIMARY clause is the name of the database server that is the default parent for copy nodes that are added to the scale-out

system. If you do not define a primary server, then you must specify the name of the parent server when you create copy nodes, so defining a primary server is recommended.

```
CREATE MIRROR SERVER "scaleout_primary_demo"
AS PRIMARY
connection_string = 'SERVER=scaleout_primary_demo;HOST=localhost:6871';
```

3. Set scale-out options for the database using the SET MIRROR OPTION statement. It is required that you specify a value for the authentication_string parameter and that you set the child_creation parameter to automatic (the default). For example:

```
SET MIRROR OPTION child_creation='automatic';
SET MIRROR OPTION authentication_string='abc';
```

Results

The root node is now ready to be used.

Next Steps

Make backup copies of the root database that are used to add child nodes to the read-only scale-out system.

Related Information

How Child Copy Nodes Are Added [page 1843]
-xp Database Option [page 563]
-x Database Server Option [page 523]
CREATE MIRROR SERVER Statement
SET MIRROR OPTION Statement

1.10.3.5.2 Adding Copy Nodes

Add copy nodes to a read-only scale-out system.

Prerequisites

This task involved backing up a copy of the database. Depending upon the backup method you choose, see the appropriate privileges for that method.

In the root database, the SET MIRROR OPTION for child_creation is set to automatic.

Context

Copy node definitions are stored in the database. You can define copy nodes for the scale-out system in advance or have the root database server define the copy nodes when they connect. It is recommended that you let the root database server create the copy node definitions because this process reduces the possibility of errors occurring in your copy node definitions.

Procedure

1. Make backup copies of the root database.
2. Start the copies of the database as copy nodes. When starting the database servers that run the copy nodes, you must specify unique names and ports, as well as the *-xp on* database option. For example:

```
dbsrv17 -n scaleout_child_demo -x TCPIP(port=6873) "c:\scaleoutdemo\copynode
\scaleoutdemo.db" -xp on
```

Results

The databases are started as copy nodes.

Related Information

How Child Copy Nodes Are Added [page 1843]
BACKUP DATABASE Statement
-xp Database Option [page 563]

1.10.3.5.3 How Child Copy Nodes Are Added

The child_creation option of the SET MIRROR OPTION statement controls how child nodes are added to a read-only scale-out system.

The following values are supported for the child_creation option:

Automatic

The root database server authenticates the copy node once it starts, or creates a new copy node if the copy node is not known. This is the recommended setting because the root server creates the definition for unknown copy nodes so that you do not have to create them manually.

Off

Connect to the root database server and execute a CREATE MIRROR SERVER statement to add a new copy node.

Manual

Add copy nodes to the tree by connecting to a copy node and executing a CREATE MIRROR SERVER statement for that database server. This statement requires MANAGE ANY MIRROR SERVER system privilege. The copy node sends a request to the root database server to define the new copy node. Once the copy node is defined, the root database server allows the new copy node to request log pages.

The database stores the connection string that is associated with the primary database server in the system. When you start a new copy node that has not been previously defined, it connects to the root database server using this connection string. The root database server uses the value of the authentication_string option that is stored in the database to authenticate the copy node.

As part of the mirror connection request, the copy node sends the copy database server's name and a string containing the copy database server IP addresses and ports to the root database server. Once the copy node is authenticated, the root database server determines whether the copy node is known. If the copy node is not known, then the root database server executes a CREATE MIRROR SERVER statement to define the new copy and its connection string. Once the copy is known, the root database server can establish a connection to it. The copy then requests all the transaction log pages that it does not already have, and once the copy node has them, the root database server starts pushing new transaction log pages to the copy node.

All copy node servers must have unique server names.

Related Information

SET MIRROR OPTION Statement
CREATE MIRROR SERVER Statement

1.10.3.5.4 Copy Node Parent Assignment

A copy node must have a parent.

The parent can be the root database server, or another copy node. You can set up the read-only scale-out system so that the parent is assigned automatically. You can also change the position of a node within the scale-out hierarchy at any time.

In this section:

Automatically Assign the Parent of a Copy Node [page 1845]
The SET MIRROR OPTION statement supports two options that can be used to assign new copy nodes to a parent in the tree so that the work of distributing transaction log pages is balanced among the nodes.

Node Positions [page 1846]
Once a copy node is running, you can assign a new parent to it by using the ALTER MIRROR SERVER statement on the root database server.

Copy Node Parent Determination [page 1846]
A copy node determines what server to connect to by looking in the ISYSMIRRORSERVER system table for its own mirror server definition and seeing what its parent value is.

1.10.3.5.4.1 Automatically Assign the Parent of a Copy Node

The SET MIRROR OPTION statement supports two options that can be used to assign new copy nodes to a parent in the tree so that the work of distributing transaction log pages is balanced among the nodes.

auto_add_server

Specifies the name of a database server that acts as the top-most node of the automatic assignment tree.

When the auto_add_server option is specified, the database server specified by that option is used as the top-most node. When the add_auto_server option is not specified, then the root database server is used as the parent for all copy nodes using automatic assignment and the auto_fan_out option is ignored.

auto_add_fan_out

Specifies the maximum number of copy nodes that each branch in the tree should have. The minimum value you can specify is 2, and the default is 10. New nodes that are created as a result of one of the following actions have their parent assigned automatically:

- connecting to the primary server when the child_creation option is set to automatic
- executing CREATE MIRROR SERVER...USING AUTO PARENT on the copy node when the child_creation option is set to manual
- executing CREATE MIRROR SERVER...USING AUTO PARENT on the root database server

The root database server determines which node in the tree should be assigned as the parent for the new copy node, based on the number of children for each node and its status as reported to the root database server by periodic status updates. The copy node sees the assignment of its new parent as it applies transaction log operations received from the root database server, so the copy node changes its parent connection when it receives the transaction log operation.

Related Information

SET MIRROR OPTION Statement
CREATE MIRROR SERVER Statement

1.10.3.5.4.2 Node Positions

Once a copy node is running, you can assign a new parent to it by using the ALTER MIRROR SERVER statement on the root database server.

This is performed by executing a statement similar to the following:

```
ALTER MIRROR SERVER "copy-server-name"
 FROM SERVER "new-parent-name";
```

The statement is recorded in the transaction log on the root database server. When the change is pushed to the copy node, the node recognizes that its own definition is being altered, and then connects to the new parent that is specified in the statement.

Related Information

ALTER MIRROR SERVER Statement

1.10.3.5.4.3 Copy Node Parent Determination

A copy node determines what server to connect to by looking in the ISYSMIRRORSERVER system table for its own mirror server definition and seeing what its parent value is.

- Null means the copy node's parent is the root node.
- If there is an ID in the parent column, then the server with that ID is the copy node's parent.

Once the copy node knows which server is its parent, it uses the connection_string option value stored in the database to connect to the correct parent.

Determining the Parent of a Copy Node in a Database Mirroring System

If scale-out is used together with a database mirroring system, copy nodes can be defined so that their partner is the server currently acting as the primary server or as the mirror server. In this configuration, one server is defined in ISYSMIRRORSERVER with type PRIMARY and one with type MIRROR. Either of these servers can be selected as a parent for a copy node.

Related Information

SYSMIRRORSERVER System View
SET MIRROR OPTION Statement

1.10.3.6 Maintaining a Read-only Scale-out System

You can check the status of the database servers in a scale-out system by monitoring the primary database using the SQL Anywhere Monitor.

You can also monitor a scale-out system from SQL Central by connecting to the primary database and checking the status of the database servers in the *Health And Statistics* pane.

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

In this section:

1.10.3.6.1 Dropping Copy Nodes from a Read-only Scale-out System

Remove a copy node database from a read-only scale-out system by deleting the mirror server definitions for the copy node.

Prerequisites

You must have the MANAGE ANY MIRROR SERVER system privilege.

Procedure

1. Connect to the primary (root) server.
2. If the copy node to be deleted has any child copy nodes, then execute an ALTER MIRROR SERVER statement to re-assign any child copy nodes to a different parent.

   ```
   ALTER MIRROR SERVER "child-copy-server-name"
      FROM SERVER "new-parent-name";
   ```
3. Execute a DROP MIRROR SERVER statement to drop the mirror server definitions from the copy node to be deleted.

   ```
   DROP MIRROR SERVER "copy-server-name";
   ```

   The mirror database stops. If the mirror database is the only database running on the server, then the server also stops.

Results

The copy node is removed from the read-only scale-out system.

Next Steps

Delete the database files.

Related Information

Copy Node Parent Assignment [page 1844]
ALTER MIRROR SERVER Statement
DROP MIRROR SERVER Statement

1.10.3.6.2 How Read-only Scale-out Systems Handle the Loss of a Parent Connection

When the parent server for a copy node becomes unavailable, the copy node's database remains available.

The copy node continues to try to connect to its parent for a period defined by the max_retry_connect_time mirror option (by default, 120 seconds). If a connection cannot be established in the specified time, then the copy attempts to connect to its alternate parent (defined by using the OR SERVER clause of the CREATE MIRROR SERVER STATEMENT) if one was defined, for an additional max_retry_connect_time seconds.

If a connection cannot be made, the copy node tries to connect to the root database server and obtain log pages from it. If the promotion_time elapses since it could not connect to its original parent or alternate parent, then the copy node requests that the root database server replaces the copy node's former parent with the copy node itself, assigning any of the copy node's siblings as its children. This behavior can result in a situation where the copy node has more children than specified by the auto_add_fan_out setting.

If the copy node is unable to connect the parent, the alternate parent, or the root database server within max_disconnected_time seconds, then the database is shut down. The max_disconnected_time default is to never shut down.

During the time the copy node is trying to connect to another database server, it continues to try to connect to its original parent. If that connection attempt is successful, the copy node resumes obtaining log pages from the original parent database server.

When a copy node is first started, it makes its database available for read-only connections even if other nodes in the tree, including its parent, are not available.

Determining the State of a Server in a Scale-Out System

The following statement to determine the state of a connection to a copy node's parent:

```
SELECT DB_EXTENDED_PROPERTY('MirrorServerState',server_name );
```

You can get an indication of how much of the transaction log has been processed on any of the nodes by executing the following statement:

```
SELECT DB_PROPERTY('CurrentRedoPos');
```

The value of the CurrentRedoPos property on a child node can be compared to the LastWrittenRedoPos database property on the primary server to get an indication of how up-to-date the copy node is.

The sa_mirror_server_status system procedure returns the connection status of all servers below the server on which the procedure is called.

Related Information

sa_mirror_server_status System Procedure
SET MIRROR OPTION Statement
CREATE MIRROR SERVER Statement

1.10.3.6.3 Converting a Partner Server to a Copy Node

Convert a partner server in a database mirroring system to a copy node in a read-only scale-out system without stopping the system.

Prerequisites

You must have the MANAGE ANY MIRROR SERVER system privilege.

Procedure

1. Connect to the partner server that you want to convert and ensure that it has the mirror role. You can only convert the partner with the mirror role. If the server you want to convert to a copy node is the primary server, you must initiate a failover so that the primary and mirror servers switch roles.
2. Connect to the primary server.
3. Change the mirror definitions:
   - Update the mirror definition of type PRIMARY to remove the connection information of the partner that is being converted.
   - Delete the mirror definition of type MIRROR.
   - Change the partner definition for the server being converted to a copy node.

   For example, execute the following statements:

   ```
   ALTER MIRROR SERVER mirror_demo_primary
   connection_string='SERVER=mirror_demo_primary;HOST=localhost:6871';

   DROP MIRROR SERVER mirror_demo_mirror;

   ALTER MIRROR SERVER mirror_server2 AS COPY FROM SERVER PRIMARY;
   ```

   Both the root server and the converted partner server restart the database as part of changing these configurations, but the server itself continues running. During the restart of the database, connections to the database are dropped
4. (Optional) If you are dismantling the mirroring system, delete the arbiter server.

Results

The partner server is now a read-only scale-out copy node.

Next Steps

Verify that the partner has been converted to a copy node by viewing the contents of the SYSMIRRORSERVER system view.

Related Information

Tutorial: Converting a Partner Server to a Copy Node [page 1860]

1.10.3.6.4 Adding a Mirroring System to a Read-only Scale-out System

Configure the root node of the read-only scale-out system to be a partner server in a mirroring system by creating an arbiter and converting a copy node to become the second partner of the mirroring system.

Prerequisites

You must have the MANAGE ANY MIRROR SERVER system privilege.

Procedure

1. Connect to the root node of the read-only scale-out system. For example, run the following command:

   ```
   dbisql -c "UID=DBA;PWD=passwd;SERVER=scaleout_root_demo"
   ```

2. Adjust the existing partner and mirror server definitions as follows:
   - Create the mirror server definition for the new mirror server.
   - Change the mirror server definition for the existing primary (root) server to include the hosts and ports of both the root server and the mirror server.
   - Change the mirror server definition for the existing partner (root) server to include a location for a state file. The state file is automatically created

   For example, execute the following statements:

   ```
   -- Define mirror server definition for the mirror server
   CREATE MIRROR SERVER scaleout_mirror_demo AS MIRROR
    connection_string='SERVER=scaleout_mirror_demo;HOST=localhost:6871,localhost:
   6873';
    -- Alter existing primary to include both partners
   ALTER MIRROR SERVER scaleout_primary_demo
   connection_string='SERVER=scaleout_primary_demo;HOST=localhost:6871,localhost:
   6873';
   ```

```
 -- Alter the existing Partner to include state file
ALTER MIRROR SERVER scaleout_root_demo state_file='c:\\scaleoutdemo\\server1\
\server1.state';
```

3. Convert an existing copy node to the mirror server. For example, execute the following statement:

```
ALTER MIRROR SERVER scaleout_child_demo AS PARTNER state_file='c:\
\scaleoutdemo\\copynode\\server3.state'
```

4. Start a server to be the arbiter server for the mirroring system. For example, run the following commands:

```
mkdir c:\scaleoutdemo\arbiter
```

```
dbsrv17 -n scaleout_arbiter_demo -su passwd -x "TCPIP(PORT=6870)" -xf "c:\
\scaleoutdemo\\arbiter\\arbiter.state" -xa "AUTH=abc;DBN=scaleoutdemo"
```

5. From the primary (root) server, add the arbiter mirror server definitions. For example, execute the following statement:

```
CREATE MIRROR SERVER scaleout_arbiter_demo
AS ARBITER
connection_string ='SERVER=scaleout_arbiter_demo;HOST=localhost:6870';
```

Results

The read-only scale-out system now is also part of a mirroring system.

Next Steps

None.

Related Information

Tutorial: Adding a Mirroring System to a Read-only Scale-out System [page 1862]
Moving a Partner Server [page 1786]

1.10.3.7 Tutorial: Creating a Read-only Scale-out System

Set up and monitor a root database server that automatically adds a copy node.

Prerequisites

You must have the MANAGE ANY MIRROR SERVER and MANAGE ANY DBSPACE system privileges.

This tutorial involves backing up a copy of the database. Depending upon the backup method you choose, see the appropriate privileges for that method.

Context

In this tutorial, all of the database servers are running on the same computer. However, each database server must be installed on a separate computer in a production environment.

If this tutorial is used with database servers running on different computers, references to localhost in the connection strings must be changed to the actual computer names.

There is a sample in *%SQLANYSAMP17%*`\SQLAnywhere\DBMirror` that uses a database mirroring system with a scale-out system.

1. Lesson 1: Creating a Read-only Scale-out System [page 1854]
   Create a read-only scale-out system, which includes a root node (the root database server) and copy nodes (read-only backup copies of the database).
2. Lesson 2: Monitoring Your Read-only Scale-out System (SQL Central) [page 1856]
   Start SQL Central and connect to `scaleoutdemo.db` to monitor your read-only scale-out system.
3. Lesson 3: Shutting Down Your Read-only Scale-out System [page 1857]
   Shut down your read-only scale-out system, which involves shutting down the scaleout_child_demo and scaleout_root_demo database servers.

Related Information

Tutorial: Adding a Mirroring System to a Read-only Scale-out System [page 1862]

1.10.3.7.1 Lesson 1: Creating a Read-only Scale-out System

Create a read-only scale-out system, which includes a root node (the root database server) and copy nodes (read-only backup copies of the database).

Prerequisites

You must have the roles and privileges listed at the beginning of this tutorial.

Context

In this tutorial, all the database servers are running on the same computer. However, each database server must be installed on a separate computer in a production environment.

If this tutorial is used with database servers running on different computers, references to localhost in the connection strings must be changed to the actual computer names.

Procedure

1. Create the directories `c:\scaleoutdemo` and `c:\scaleoutdemo\copynode`.
2. At a command prompt, run the following command to create the database `scaleoutdemo.db`, which contains data from the sample database:

   ```
   newdemo.bat c:\scaleoutdemo\scaleoutdemo.db
   ```
3. Start the root database server for the scale-out system:

   ```
   dbsrv17 -n scaleout_root_demo -su passwd -x TCPIP(port=6871) "c:\scaleoutdemo
   \scaleoutdemo.db" -xp on
   ```

   It is recommended that you include the -su option to specify the password for the utility database so that you can connect to the utility database to shut down the database server, if necessary.
4. Connect to the database from Interactive SQL:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=scaleout_root_demo;DBN=scaleoutdemo"
   ```
5. In Interactive SQL, define the root database server of the scale-out system:

   ```
   CREATE MIRROR SERVER "scaleout_primary_demo"
   AS PRIMARY
   connection_string = 'SERVER=scaleout_primary_demo;HOST=localhost:6871';
   ```
6. Define the root database server as a partner in the scale-out system. The name of the partner server must match the database server name that is used in the command to start the database server.

   ```
   CREATE MIRROR SERVER "scaleout_root_demo"
   ```

```
AS PARTNER
connection_string = 'SERVER=scaleout_root_demo;HOST=localhost:6871';
```

7. Set the options for the root database server of the scale-out system:

```
SET MIRROR OPTION auto_add_server='scaleout_primary_demo';
SET MIRROR OPTION child_creation='automatic';
SET MIRROR OPTION authentication_string='abc';
SET MIRROR OPTION auto_add_fan_out='10';
```

8. Make a backup copy of the database, placing it in the directory `c:\scaleoutdemo\copynode`.

```
BACKUP DATABASE DIRECTORY 'c:\\scaleoutdemo\\copynode';
```

9. At a command prompt, start the backup copy of the database as a copy node of the scaleout_root_demo root database server:

```
dbsrv17 -n scaleout_child_demo -su passwd -x TCPIP(port=6873) "c:\scaleoutdemo
\copynode\scaleoutdemo.db" -xp on
```

10. Connect to the copy node from Interactive SQL:

```
dbisql -c "UID=DBA;PWD=sql;SERVER=scaleout_child_demo"
```

    Once the copy node connects to the root database server, you are warned that the copy node is a read-only copy of the database. You can now connect to the copy node and execute queries against it.

11. In Interactive SQL, you can view the mirror servers in the scale-out system by running the following query:

```
SELECT * FROM SYSMIRRORSERVER;
```

12. Close all Interactive SQL windows.

Results

You have created a read-only scale-out system.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Creating a Read-only Scale-out System [page 1853]

Next task: Lesson 2: Monitoring Your Read-only Scale-out System (SQL Central) [page 1856]

Related Information

How Scale-out Works [page 1832]

Read-only Scale-out Setup [page 1839]
Maintaining a Read-only Scale-out System [page 1847]
-su Database Server Option [page 502]

1.10.3.7.2 Lesson 2: Monitoring Your Read-only Scale-out System (SQL Central)

Start SQL Central and connect to `scaleoutdemo.db` to monitor your read-only scale-out system.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Context

> **i** Note
>
> Adobe will stop updating and distributing the Flash Player at the end of 2020. Because the SQL Anywhere Monitor is based on Flash, you cannot use it once Flash support ends. In many cases, tasks that were previously performed in the Monitor can be performed in the SQL Anywhere Cockpit. See SQL Anywhere Monitor Non-GUI User Guide.

Procedure

1. In SQL Central, open the *Connect* window, and complete the following fields to connect to the test database, `scaleoutdemo.db`:
   a. In the *User ID* field, type **`DBA`**.
   b. In the *Password* field, type **`sql`**.
   c. In the *Action* field, select *Connect to a running database on this computer*.
   d. In the *Server Name* field, type **`scaleout_root_demo`**.
   e. Click *Connect*.
2. Click the *Overview* tab in the main window to monitor the database `scaleoutdemo.db`.
3. Click the *Database Mirroring and Scale-out* dropdown list to view the scale-out system's primary server name, as well as information about the scale-out system's copy nodes.

Results

You have connected to and monitored your read-only scale-out system using SQL Central.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Creating a Read-only Scale-out System [page 1853]

Previous task: Lesson 1: Creating a Read-only Scale-out System [page 1854]

Next task: Lesson 3: Shutting Down Your Read-only Scale-out System [page 1857]

Related Information

SQL Anywhere Monitor [page 1265]
Quick Start to Using the Monitor [page 1270]

1.10.3.7.3 Lesson 3: Shutting Down Your Read-only Scale-out System

Shut down your read-only scale-out system, which involves shutting down the scaleout_child_demo and scaleout_root_demo database servers.

Prerequisites

You must have completed the previous lessons in this tutorial.

You must have the roles and privileges listed at the beginning of this tutorial.

Procedure

Shut down your read-only scale-out system.

a. Double-click the network server icon in the system tray for the scaleout_child_demo database server.

b. Click *Shut Down* in the database server messages window.
c. Click *Yes* to shut down the database server.
d. Repeat this step to shut down the scaleout_root_demo database server.

Results

The read-only scale-out system is shut down.

Task overview: Tutorial: Creating a Read-only Scale-out System [page 1853]

Previous task: Lesson 2: Monitoring Your Read-only Scale-out System (SQL Central) [page 1856]

1.10.3.8 Tutorial: Using One Server as Both a Copy Node and an Arbiter

Set up a server to run as both the arbiter server and as a copy node.

Prerequisites

You must have the MANAGE ANY MIRROR SERVER system privilege.

This task involved backing up a copy of the database. Depending upon the backup method you choose, see the appropriate privileges for that method.

Context

This tutorial describes how to set up a mirroring system that is also involved in read-only scale-out.

Although it is recommended that the arbiter server in a mirroring system run on a physically separate computer from the other servers, it is not always feasible.

In this tutorial, all the database servers are running on the same computer. However, each database server must be installed on a separate computer in a production environment.

If this tutorial is used with database servers running on different computers, references to localhost in the connection strings must be changed to the actual computer names.

Procedure

1. Create the following directories: `c:\server1`, `c:\server2`, and `c:\arbiter`.
2. Create a database named `mirror_demo.db` that contains data from the sample database and has a transaction log. You cannot start a database that doesn't have a transaction log in mirroring mode. Run the following command:

   ```
   newdemo.bat c:\server1\mirror_demo.db
   ```

3. Start the first database server. Run the following command:

   ```
   dbsrv17 -n mirror_server1 -x "tcpip(PORT=6871)" -su passwd "c:
   \server1\mirror_demo.db" -xp on
   ```

4. Connect to the mirror_demo database. For example, run the following command:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_server1"
   ```

5. Create the mirror server definitions. For example, execute the following SQL statements:

   ```
   CREATE MIRROR SERVER mirror_demo_primary AS PRIMARY
   connection_string='SERVER=mirror_demo_primary;HOST=localhost:6871,localhost:
   6872';
   CREATE MIRROR SERVER mirror_demo_mirror AS MIRROR
   connection_string='SERVER=mirror_demo_primary;HOST=localhost:6871,localhost:
   6872';

   CREATE MIRROR SERVER mirror_server1 AS PARTNER
   connection_string='SERVER=mirror_server1;HOST=localhost:6871' state_file='c:
   \server1\\server1.state';

   CREATE MIRROR SERVER mirror_server2 AS PARTNER
   connection_string='SERVER=mirror_server2;HOST=localhost:6872' state_file='c:
   \server2\\server2.state';
   ```

6. Create the arbiter server definition with the CREATE MIRROR SERVER statement. The `mirror-server-name` for the arbiter must be different from the server name specified in the connection string.

   ```
   CREATE MIRROR SERVER demo_arbiter AS ARBITER
    connection_string='SERVER=demo_server3;HOST=localhost:6870';
   ```

7. Set mirroring options for the mirroring system. You must specify an authentication string and set the auto_add_server option to the name of the primary server. Execute the following statements:

   ```
   SET MIRROR OPTION auto_add_server='mirror_demo_primary';
   SET MIRROR OPTION authentication_string='abc';
   ```

   In the arbiter server definition, the `arbiter-server-name` must be different from the actual server name for the arbiter. The client connection string for the arbiter must include the ServerName [SERVER] connection parameter with the actual server name.

8. Create the mirror server.
   a. Make copies of the database file and transaction log in `c:\server1`, and add them to `c:\server2` and `c:\arbiter`. Run the following command:

      ```
      dbbackup -c "UID=DBA;PWD=sql;SERVER=mirror_server1;DBN=mirror_demo" server2
      dbbackup -c "UID=DBA;PWD=sql;SERVER=mirror_server1;DBN=mirror_demo" arbiter
      ```

b. Start the mirror server:

```
dbsrv17 -n mirror_server2 -x "tcpip(port=6872)" server2\mirror_demo.db -su
passwd -xp on
```

9. Start the arbiter server as a copy node and as an arbiter by specifying the -xa, -xp, and -xf server options.

```
dbsrv17 -n demo_server3 -x "tcpip(port=6870)" arbiter\mirror_demo.db -xp on -
xf arbiter\arbiter.state -xa "AUTH=abc;DBN=mirror_demo" -su passwd
```

10. Use the dbping utility to determine when the HA configuration is ready:

```
 dbping -c "UID=DBA;PWD=sql;SERVER=mirror_demo_primary"  -pd
Mirrorstate,PartnerState,ArbiterState
```

```
SQL Anywhere Server Ping Utility Version 17.0.11.1293
Type       Property                Value
---------  ----------------        ------------------------------
Database   MirrorState             synchronized
Database   PartnerState            connected
Database   ArbiterState            connected
Ping database successful.
```

11. Use the following query to shows the nodes being mirrored in this HA system:

```
dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_server1" "SELECT server_name,state
FROM sa_mirror_server_status()"
```

12. Verify that demo_server3 is a COPY node and that the arbiter is connected by inspecting the contents of the SYSMIRRORSERVER table:

```
dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_demo_primary" "SELECT  server_name,
server_type FROM SYSMIRRORSERVER"
```

Results

The server that runs the arbiter also runs a copy node.

1.10.3.9 Tutorial: Converting a Partner Server to a Copy Node

Convert a partner server in a database mirroring system to a copy node in a read-only scale-out system without stopping the system.

Prerequisites

This tutorial relies on the database mirroring system described in the tutorial on creating a database mirroring system.

You must have the MANAGE ANY MIRROR SERVER system privilege.

By default, to use the ALTER DATABASE...SET PARTNER FAILOVER statement you must have the SERVER OPERATOR system privilege. The required privileges can be changed by using the -gd database server option.

Context

In this tutorial, all the database servers are running on the same computer. However, each database server must be installed on a separate computer in a production environment.

If this tutorial is used with database servers running on different computers, references to localhost in the connection strings must be changed to the actual computer names.

Procedure

1. Connect to the partner server that you want to convert and ensure that it has the mirror role. You can only convert the partner with the mirror role. If the server you want to convert to a copy node is the primary server, you must initiate a failover so that the primary and mirror servers switch roles. For example, connect to mirror_server2:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_server2;HOST=localhost:6872"
   ```

   The database server property MirrorRole returns the current role of the server. Execute the following statement:

   ```
   SELECT DB_PROPERTY( 'MirrorRole' );
   ```

   If Primary is returned, then initiate a failover by executing the following statement:

   ```
   ALTER DATABASE SET PARTNER FAILOVER;
   ```

   The current primary database stops and restarts.
2. Connect to the primary server:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_demo_primary;HOST=localhost:
   6871,localhost:6872"
   ```
3. Change the definitions:
   - Update the mirror definition of type PRIMARY to remove the connection information of the partner that is being converted.
   - Delete the mirror definition of type MIRROR.
   - Change the partner definition for the server being converted to a copy node.

   Execute the following statements:

   ```
   ALTER MIRROR SERVER mirror_demo_primary
   connection_string='SERVER=mirror_demo_primary;HOST=localhost:6871';

   DROP MIRROR SERVER mirror_demo_mirror;

   ALTER MIRROR SERVER mirror_server2 AS COPY FROM SERVER PRIMARY;
   ```

The database on both the partner servers restart the database as part of changing these configurations, but the server itself continues running. During the database restart, connections to the database are dropped.

4. (Optional) If you are dismantling the mirroring system, delete the arbiter server. Execute the following statement:

```
DROP MIRROR SERVER demo_arbiter;
```

5. Verify that mirror_server2 has been converted to a copy node by viewing the contents of the SYSMIRRORSERVER system view. Run the following command:

```
dbisql -c "UID=DBA;PWD=sql;SERVER=mirror_demo_primary" "SELECT server_name,
server_type FROM SYSMIRRORSERVER"
```

Results

The mirror server is now a read-only scale-out copy node.

Related Information

Tutorial: Creating a Database Mirroring System [page 1800]

1.10.3.10 Tutorial: Adding a Mirroring System to a Read-only Scale-out System

Add a mirroring system to an existing read-only scale-out system. The root of the read-only scale-out system also becomes the mirror server.

Prerequisites

This tutorial assumes that you have a running read-only scale-out system as described in the tutorial on creating a read-only scale-out system.

Configure the root node of the read-only scale-out system to be the primary server in a mirroring system by defining mirror servers, creating an arbiter, and converting copy nodes to become the mirror partner of the mirroring system

You must have the MANAGE ANY MIRROR SERVER system privilege.

This tutorial involves backing up a copy of the database. Depending upon the backup method you choose, see the appropriate privileges for that method.

Context

In this tutorial, all the database servers are running on the same computer. However, each database server must be installed on a separate computer in a production environment.

If this tutorial is used with database servers running on different computers, references to localhost in the connection strings must be changed to the actual computer names.

Procedure

1. Connect to the root node of the read-only scale-out system. Run the following command:

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=scaleout_root_demo"
   ```

2. Define the primary and mirror servers.
   a. Create the directory `c:\scaleoutdemo\server1`.
   b. Execute the following statements:

   ```
   -- Alter existing primary to include both partners
   ALTER MIRROR SERVER scaleout_primary_demo
   connection_string='SERVER=scaleout_primary_demo;HOST=localhost:
   6871,localhost:6873';
   -- Alter existing Partner to include state file
   ALTER MIRROR SERVER scaleout_root_demo state_file='c:\\scaleoutdemo\
   \server1\\server1.state';
   -- Define mirror
   CREATE MIRROR SERVER scaleout_mirror_demo AS MIRROR
   connection_string='SERVER=scaleout_mirror_demo;HOST=localhost:
   6871,localhost:6873';
   --Convert an existing copy node to the mirror server
   ALTER MIRROR SERVER scaleout_child_demo
   connection_string='SERVER=scaleout_child_demo;HOST=localhost:6873'
   state_file='c:\\scaleoutdemo\\copynode\\server3.state';
   ```

3. Convert an existing copy node to the mirror server. Execute the following statement:

   ```
   --Convert an existing copy node to the mirror server
   ALTER MIRROR SERVER scaleout_child_demo
   connection_string='SERVER=scaleout_child_demo;HOST=localhost:6873'
   state_file='c:\\scaleoutdemo\\copynode\\server3.state';
   ```

4. Start a server to be the arbiter server for the mirroring system. Run the following commands:

   ```
   mkdir c:\scaleoutdemo\arbiter
   ```

   ```
   dbsrv17 -n scaleout_arbiter_demo -su passwd -x "TCPIP(PORT=6870)" -xf "c:\
   \scaleoutdemo\\arbiter\\arbiter.state" -xa "AUTH=abc;DBN=scaleoutdemo"
   ```

5. Connect to the root node and add the arbiter. Execute the following statements.

   ```
   dbisql -c "UID=DBA;PWD=sql;SERVER=scaleout_primary_demo"
   ```

   ```
   CREATE MIRROR SERVER scaleout_arbiter_demo
   AS ARBITER
   connection_string ='SERVER=scaleout_arbiter_demo;HOST=localhost:6870';
   ```

6. Convert an existing copy node to the mirror server. Execute the following statement:

```
ALTER MIRROR SERVER scaleout_child_demo AS PARTNER state_file='c:\
\scaleoutdemo\\copynode\\server3.state'
```

7. Verify that scaleout_child_demo is the mirror server by inspecting the contents of the SYSMIRRORSERVER table. Run the following command:

```
dbisql -c "UID=DBA;PWD=sql;SERVER=scaleout_primary_demo" "SELECT
server_name, server_type FROM SYSMIRRORSERVER"
```

   The scaleout_child_demo server is now the second PARTNER server.

8. Update the connection strings in your client applications to specify the addresses of both partners in the Host connection parameter. For example:

```
dbping -c "UID=DBA;PWD=sql;SERVER=scaleout_primary_demo;HOST=localhost:
6871,localhost:6873"
```

9. (Optional) Add another copy node.
   a. Make a backup copy of the primary database, placing it in the directory `c:\scaleoutdemo\copynode2`.
   b. Execute the following statement:

```
BACKUP DATABASE DIRECTORY 'c:\\scaleoutdemo\\copynode2';
```

   c. Start the backup copy of the database as a child (copy node) of the scaleout_root_demo database server. Run the following command:

```
dbsrv17 -n scaleout_child2_demo -su passwd -x TCPIP(port=6874) "c:
\scaleoutdemo\copynode2\scaleoutdemo.db" -xp on
```

Results

The read-only scale-out system now is also part of a mirroring system.

Related Information

Tutorial: Creating a Read-only Scale-out System [page 1853]
Adding a Mirroring System to a Read-only Scale-out System [page 1851]

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon  : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon  : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

THE BEST RUN SAP