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

MobiLink Getting Started

THE BEST RUN

Content

1 MobiLink - Getting Started

This book describes MobiLink, a session-based relational-database synchronization system. MobiLink technology allows two-way replication and is well suited to mobile computing environments.

In this section:

1.1 MobiLink Synchronization

MobiLink is a session-based synchronization technology designed to synchronize UltraLite and SQL Anywhere remote databases with a consolidated database.

In this section:

There are two basic architectures for database applications: online applications and occasionally connected smart client applications.

MobiLink Application Development Options [page 16]
MobiLink provides a variety of ways to develop an application. You can use these methods alone or in combination.

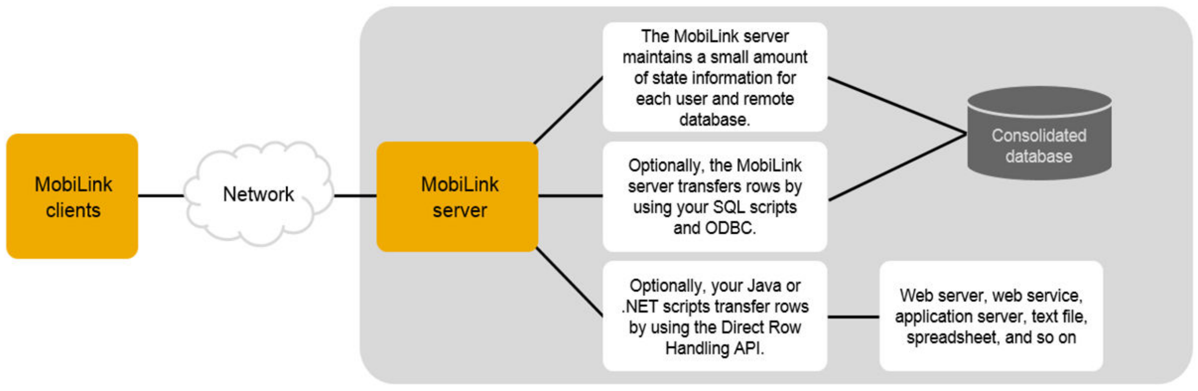
Options for Writing Server-side Synchronization Logic [page 16]
MobiLink synchronization scripts can be written in SQL, or they can be written in Java (using the MobiLink server API for Java) or in .NET (using the MobiLink server API for .NET).

The Synchronization Process [page 18]
A **synchronization** is a process of data exchange between MobiLink clients and a central data source. During this process, the client must establish and maintain a session with the MobiLink server. If successful, the session leaves the remote and consolidated databases in a mutually consistent state.

MobiLink Security Considerations [page 26]
There are several aspects to securing data throughout a widely distributed system such as a MobiLink installation.

1.1.1 Parts of a MobiLink Application

In MobiLink synchronization, many clients synchronize through the MobiLink server to central data sources.


MobiLink clients
Network
MobiLink server
The MobiLink server maintains a small amount of state information for each user and remote database.
Optionally, the MobiLink server transfers rows by using your SQL scripts and ODBC.
Optionally, your Java or .NET scripts transfer rows by using the Direct Row Handling API.
Consolidated database
Web server, web service, application server, text file, spreadsheet, and so on

MobiLink clients

The client can be installed on an Android device, a server, or desktop computer. Two types of clients are supported: UltraLite and SQL Anywhere databases. Either or both can be used in a MobiLink installation.

Network

The connection between the MobiLink server and the MobiLink client can use several protocols. See:

- MobiLink server: -x mlsrv17 option
- UltraLite and SQL Anywhere clients: MobiLink client network protocol options

MobiLink server

This server manages the synchronization process and provides the interface between all MobiLink clients and the consolidated database server.

Consolidated database

This database typically contains the central copy of your application information in the synchronization system. It also typically holds system tables and procedures that are required by MobiLink synchronization, and state information needed to synchronize.

State information

The MobiLink server typically maintains synchronization information in system tables in the consolidated database. It does this over an ODBC connection.

You can also choose to store your state information in a separate database.

SQL row handling

If you provide the MobiLink server with SQL scripts, it uses these scripts to transfer rows to and from the consolidated database over an ODBC connection.

Direct row handling

In addition to a consolidated database, you can optionally synchronize with other data sources using MobiLink direct row handling.

Synchronization scripts

You write synchronization scripts for each table in the remote database and you save these scripts in MobiLink system tables in the consolidated database. These scripts determine what is done with the uploaded data, and what data to download. There are two types of script: table scripts and connection-level scripts.

Related Information

MobiLink Server
MobiLink Clients
MobiLink System Database
MobiLink Consolidated Databases
Overview of MobiLink Events
Synchronization Scripts
Synchronization Events
Options for Writing Server-side Synchronization Logic [page 16]
Direct Row Handling
-x mlsrv17 Option
MobiLink Client Network Protocol Options

1.1.2 MobiLink Features and Architecture

MobiLink synchronization is adaptable and flexible.

The following are some of its key features:

Features

Easy to get started

Using the *Create Synchronization Model Wizard*, you can create synchronization applications quickly. The wizard can handle many difficult implementation details of complex synchronization systems. SQL Central allows you to view a synchronization model offline, provides an easy interface for making changes, and has a deployment option for you to deploy the model to your consolidated database.

Monitoring and reporting

MobiLink provides three mechanisms for monitoring your synchronizations: the MobiLink Profiler, the SQL Anywhere Monitor for MobiLink, and statistical scripts.

> 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.

Performance tuning

There are several mechanisms for tuning MobiLink performance. For example, you can adjust the degree of contention, upload cache size, number of database connections, logging verbosity, or BLOB cache size.

Scalability

MobiLink is an extremely scalable and robust synchronization platform. A single MobiLink server can handle thousands of simultaneous synchronizations, and multiple MobiLink servers can be run simultaneously using load balancing. The MobiLink server is multithreaded and uses connection pooling with the consolidated database.

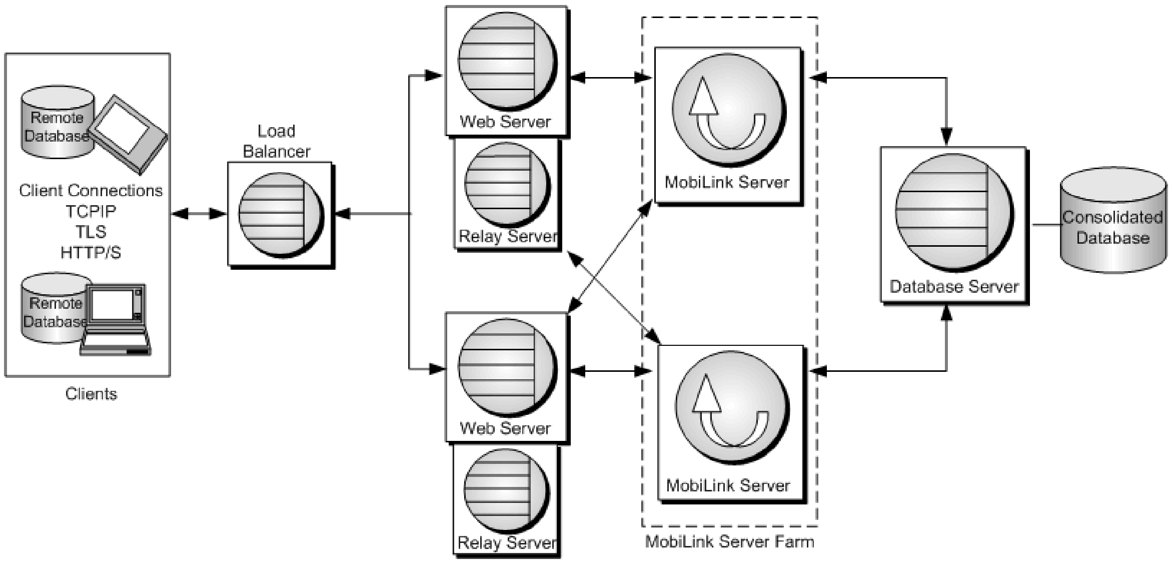
Security

MobiLink provides extensive security options, including user authentication that can be integrated with your existing authentication, encryption, and transport layer security that works by the exchange of secure certificates. MobiLink also provides FIPS-certified security options.

Relay Server reverse proxy

The Relay Server is a reverse proxy that enables secure, load-balanced communication between mobile devices and backend servers through a web server.

The diagram below shows how the Relay Server fits into a MobiLink environment.



Architecture

Data coordination

MobiLink allows you to choose selected portions of the data for synchronization. MobiLink synchronization also allows you to resolve conflicts between changes made in different databases. The synchronization process is controlled by synchronization logic, which can be written as a SQL, Java, or .NET application. Each piece of logic is called a **script**. With scripts, for example, you can specify how uploaded data is applied to the consolidated database, specify what gets downloaded, and handle different schema and names between the consolidated and remote databases. Event-based scripting provides great flexibility in the design of the synchronization process, including such features as conflict resolution, error reporting, and user authentication.

Two-way synchronization

Changes to a database can be made at any location.

Upload-only or download-only synchronization

By default synchronization is two-way, with both an upload and a download. However, you can also choose to perform an upload-only synchronization or a download-only synchronization.

File-based download

Downloads can be distributed as files, enabling offline distribution of synchronization changes. This feature includes functionality to ensure that the correct data is applied.

Server-initiated synchronization

You can initiate MobiLink synchronization from the consolidated database. This means you can push data updates to remote databases, and cause remote databases to upload data to the consolidated database.

You can use server-initiated remote tasks (SIRT) as an alternative to server-initiated synchronization.

Choice of network protocols

Synchronization can occur over TCP/IP, HTTP, or HTTPS.

Session-based

All changes can be uploaded in a single transaction and downloaded in a single transaction. At the end of each successful synchronization, the consolidated and remote databases are consistent. (To preserve the order of transactions, you can also choose to have each transaction on the remote database uploaded as a separate transaction.)

Either a whole transaction is synchronized, or none of it is synchronized. This ensures transactional integrity for each database.

Data consistency

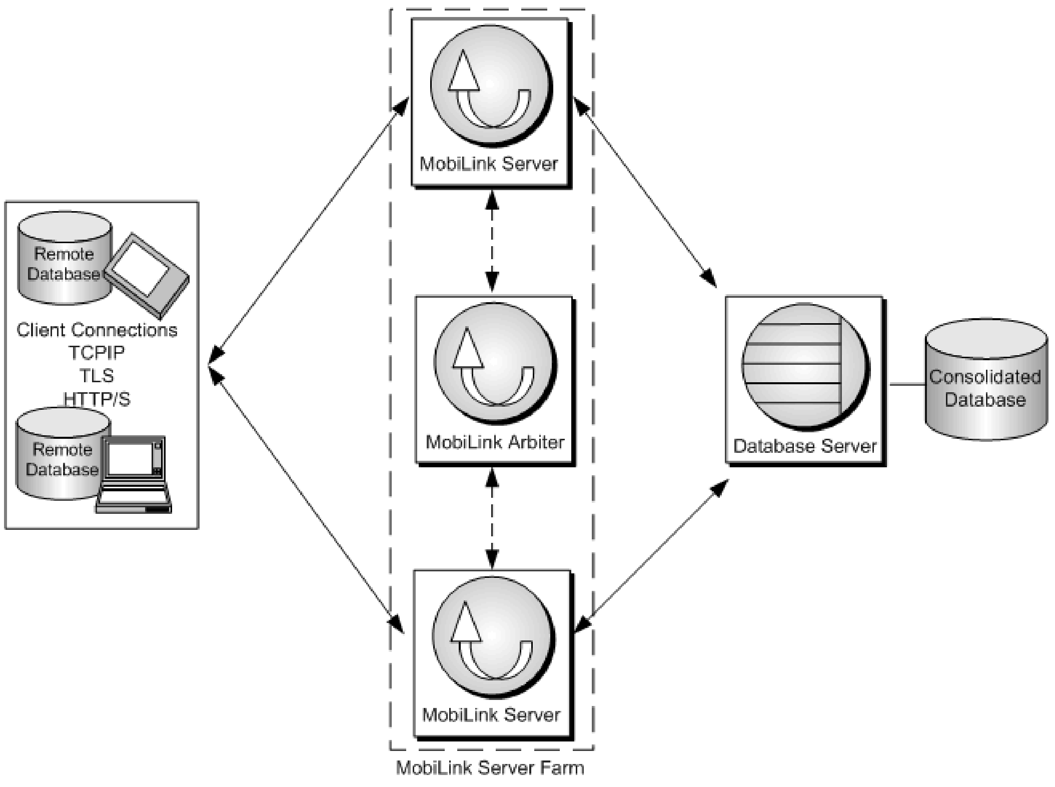
MobiLink operates using a loose consistency policy. All changes are synchronized with each site over time in a consistent manner, but different sites may have different copies of data at any instant.

Wide variety of hardware and software platforms

A variety of widely used database management systems can be used as a MobiLink consolidated database, or you can define synchronization to an arbitrary data source using the MobiLink server API. Remote databases can be SQL Anywhere or UltraLite. The MobiLink server runs on Microsoft Windows, UNIX (including macOS), and Linux. SQL Anywhere runs on Microsoft Windows, UNIX (including macOS), and Linux. UltraLite runs on Android, iOS, Microsoft Windows Phone, and Microsoft Windows Mobile.

MobiLink arbiter

A MobiLink arbiter ensures that only a single MobiLink server in a server farm is running as the primary server. This prevents redundant notifications in a server-initiated synchronization environment. The diagram below shows the MobiLink arbiter in a server farm environment.



Related Information

Introduction to the Relay Server
Manage Remote Databases
Supported Platforms
Server-initiated Remote Tasks (SIRT)

1.1.3 Quick Start to MobiLink

MobiLink is designed to synchronize data among many remote applications that connect intermittently with one or more central data sources.

In a basic MobiLink application, your remote clients are SQL Anywhere or UltraLite databases, and your central data source is one of the supported ODBC-compliant relational databases. This architecture can be extended using the MobiLink server API so that there are virtually no restrictions on what you synchronize to on the server side.

In all MobiLink applications, the MobiLink server is the key to the synchronization process. Synchronization typically begins when a MobiLink remote site opens a connection to a MobiLink server. During synchronization, the MobiLink client at the remote site can upload database changes that were made to the remote database since the previous synchronization. On receiving this data, the MobiLink server updates the consolidated database, and then can download changes from the consolidated database to the remote database.

The quickest way to start developing a MobiLink application is to use the *Create Synchronization Model Wizard*. When you use the wizard, most of the steps outlined below are handled for you.

However, even when using a MobiLink model, you need to understand the process and components of MobiLink synchronization.

Overview of a MobiLink Application

- Set up a consolidated database
  Run setup scripts against the database to add system objects required by MobiLink synchronization. Alternatively, you can create a separate system database to hold these objects.
- Set up remote databases
  - Your remote databases can be SQL Anywhere, UltraLite, or a combination of the two.
  - In your remote databases, create MobiLink users.
  - To determine the upload in a SQL Anywhere remote database, create publications and subscriptions. To determine the upload in an UltraLite remote database, create publications.
- Create server synchronization logic to determine how the upload is applied.
- Set up timestamp-based synchronization to download data that has changed since the last download.
- Start the MobiLink server.
- Initiate synchronization on the client.

Introductory Reading

Refer to the MobiLink - Getting Started book for introductory information about MobiLink.

The MobiLink- Server Administration book also contains information to help you get started with MobiLink, including topics on synchronization techniques and writing scripts.

Tutorials

The following tutorials range from introductory tutorials for news users to demonstrations of how to use advanced features.

- CustDB sample for MobiLink
- MobiLink Contact sample
- Tutorial: Introducing MobiLink
- Tutorial: Using MobiLink with a SQL Anywhere consolidated database
- Tutorial: Building the UltraLite CustDB sample application
- Tutorial: Using MobiLink with an Oracle Database 11g
- Tutorial: Using MobiLink with an Adaptive Server Enterprise consolidated database
- Tutorial: Using Java or .NET for custom user authentication
- Tutorial: Using direct row handling
- Tutorial: Synchronizing with Microsoft Excel
- Tutorial: Synchronizing with XML
- Tutorial: Using central administration of remote databases
- Tutorial: Changing a schema using the script version clause
- Tutorial: Changing a schema using the ScriptVersion extended option
- Tutorial: Simulating multiple MobiLink clients using the MobiLink Replay utility

Other Resources for Getting Started

- MobiLink provides samples that you can examine and run to explore MobiLink functionality. MobiLink samples are installed with the product in *%SQLANYSAMP17%*`\MobiLink`.
- The SAP SQL Anywhere Community has links to blogs maintained by executives, employees, and developers who exchange views and ideas about using SQL Anywhere, MobiLink, and related technologies.

Related Information

Synchronization Models [page 37]
MobiLink Consolidated Databases
MobiLink Users in a Synchronization System

Publications
Synchronization Scripts
MobiLink Server
Initiation of Synchronization
UltraLite Client Synchronization Design
Synchronization Techniques
CustDB Sample for MobiLink [page 71]
MobiLink Contact Sample [page 90]
Publishing Data in UltraLite
Implementing Timestamp-based Downloads
Tutorial: Introducing MobiLink [page 106]
Tutorial: Using MobiLink with a SQL Anywhere Consolidated Database [page 129]
Tutorial: Building the UltraLite CustDB Sample Application
Tutorial: Using MobiLink with an Oracle Database 11g [page 143]
Tutorial: Using MobiLink with an Adaptive Server Enterprise Consolidated Database [page 165]
Tutorial: Using Java or .NET for Custom User Authentication [page 188]
Tutorial: Using Direct Row Handling [page 198]
Tutorial: Synchronizing with Microsoft Excel [page 225]
Tutorial: Synchronizing with XML [page 247]
Tutorial: Using Central Administration of Remote Databases [page 269]
Tutorial: Changing a Schema Using the Script Version Clause [page 300]
Tutorial: Changing a Schema Using the ScriptVersion Extended Option [page 311]
Tutorial: Simulating Multiple MobiLink Clients Using the MobiLink Replay Utility [page 319]
SAP community network

1.1.4 MobiLink Application Design

There are two basic architectures for database applications: online applications and occasionally connected smart client applications.

Online applications

Users update data by connecting to the central database directly. When a connection is unavailable, the user cannot work.

Occasionally connected smart client applications

Each user has a local database. Their database application is always available to them, regardless of connectivity, and is kept synchronized with other databases in the system.

MobiLink is designed for creating occasionally connected smart client applications. Smart client applications can greatly increase the usability, efficiency, and scalability of an application, but they pose new issues for application developers. Some of the major issues facing developers of smart client applications, and how you can implement solutions in a MobiLink synchronization environment are described below.

Synchronize Only What You Need

In most applications it would be a disaster to download the entire consolidated database every time you want to update any piece of data on your remote device. The time and bandwidth would be prohibitive, making the whole system unworkable. There are various techniques for ensuring you upload and download only what each user needs.

First, each remote database should only contain a subset of the tables and columns in the consolidated database. For example, a salesperson in Region A may need different tables and columns than a salesperson in Region B or a supervisor.

Of the tables and columns that you put on a remote device, you only want to mark ones for synchronization that need to be synchronized. In a MobiLink application you can map tables and columns, regardless of their names, as long as the data types match. By default, data is both uploaded and downloaded, but MobiLink also allows you to specify that certain columns are upload-only or download-only.

Your synchronization should only download rows to a remote database that are relevant to the user. You might want to partition your download by remote database, by user, or by other criteria. For example, a sales rep in Region A may only need data updates about Region A.

You only want to update data that has changed. In a MobiLink application the upload is based on the transaction log and so by default, data is only uploaded if it has changed on the remote database. To do the same for the download, you specify timestamp-based synchronization so that your system records the time that data is successfully downloaded, and data is downloaded only if it has changed since then.

You may also want to implement a system of high priority synchronization: time-sensitive data is scheduled to be updated frequently, but less time-critical data is scheduled to be updated at night or when the device is in a cradle. You can implement high-priority synchronization by creating different publications that are scheduled to run at different times.

In addition, your users may benefit from a push-synchronization system, where data is effectively pushed down to remote devices when needed. For example, if a trucking company dispatcher learns of a traffic disruption, they can download an update to the truck drivers who are heading towards that area. In MobiLink, this is called server-initiated synchronization.

Handle Upload Conflicts

Say you have a warehouse. Each employee has a mobile device that they use to update inventory as they add or remove boxes. They start a shift with 100 boxes, so each employee's remote database registers 100, as does the consolidated database. David removes 20 boxes. He updates his database and synchronizes. Now both his database and the consolidated database register 80. Now Susan removes ten boxes. But when Susan updates her database and synchronizes, her application expects the consolidated database to have 100 boxes, not 80. This generates an upload conflict.

In this warehouse application, the solution is to create conflict resolution logic that says that the correct value is whatever David updated it to, minus the original value less Susan's value:

```
80 - (100 - 90) = 70
```

While this conflict resolution logic works for inventory-based applications such as a warehouse, it isn't appropriate in all business applications. With MobiLink, you can define conflict resolution logic to cover:

Inventory model

Update the row for the correct number of units.

Date

The latest update wins (based on when the value was changed in the database, not when the value was synchronized).

Person

For example, the manager always wins or the owner of the record always wins.

Custom

Just about any other business logic you need to implement.

Sometimes you can design your system so that upload conflicts cannot occur. If data is partitioned on the remotes so that there is no overlap, conflicts may be avoided. However, if conflicts can happen, you should create a programmatic solution for detecting and resolving them.

Unique Primary Keys

To upload data, detect upload conflicts, and synchronize deleted rows on the consolidated database, you must have unique primary keys on every synchronized table in your database system. Each row must have a primary key that is unique not only within the database, but within the entire database system. Primary keys must not be updated.

MobiLink provides several ways to guarantee unique primary keys. One is to set the data type of the primary key to a GUID. GUID, which stands for Globally Unique Identifier, is a 16-byte hexadecimal number. MobiLink provides a NEWID function that causes a GUID to be created automatically for a new row.

Another solution is a composite key. In MobiLink, each remote database has a unique value called a remote ID. Your primary keys could be formed from the remote ID plus a regular primary key, such as an ordinal value.

SQL Anywhere also offers a global autoincrement solution. You declare a column as GLOBAL AUTOINCREMENT and then when a row is added, the primary key is automatically created by incrementing the last value. This solution works best when your consolidated database is SQL Anywhere.

Finally, you can create a pool of primary key values that are distributed to remote databases.

How you choose which primary key system to use, like many decisions in developing a synchronization solution, has to do with the level of control you have over the consolidated and remote databases. Often, the remote databases must be able to operate without any administration. You may also find that it is difficult to change the schema on the consolidated database. In addition, your choice of RDBMS for the consolidated database may limit your options, as not all RDBMSs support all features.

Handling Deletes

Another issue in a synchronization system is how to handle rows that are deleted from the consolidated database. Say I delete a row from the consolidated database. The next time David synchronizes his remote database, the delete is downloaded, deleting the row from David's database. But what do I do with it on the consolidated database? I can't delete it because I need to download the delete to Susan as well.

Here are two ways you can handle download deletes. First, you can add a status column to each table that indicates whether the row is deleted or not. In this case, the row is never deleted, it is just marked for deletion. You can occasionally clean up the rows marked for deletion, once you are sure that all the remote databases are up to date. Alternatively, you can create a shadow table for each table. The shadow table stores the primary key values of deleted rows. When a row is deleted, a trigger populates the shadow table, and the values in the shadow table determine what to delete on the remote database.

Transactions

In a synchronized database system, only database transactions that are committed should be synchronized. In addition, all committed transactions involving data that is to be synchronized should be synchronized, or an error should be generated. This is the default behavior in MobiLink.

You also need to consider the isolation level of the connection to the consolidated database. You need to use an isolation level that provides the best performance possible while ensuring data consistency. Isolation level 0 (READ UNCOMMITTED) is generally unsuitable for synchronization as it can lead to inconsistent data.

By default, MobiLink uses the isolation level SQL_TXN_READ_COMMITTED for uploads, and if possible it uses snapshot isolation for downloads (otherwise it uses SQL_TXN_READ_COMMITTED). Snapshot isolation eliminates the problem of downloads being blocked until transactions are closed on the consolidated database, but not all RDBMSs support it.

Daylight Savings Time

The annual change to daylight savings time can pose a problem for synchronized databases during the hour that the time changes. In the autumn the time moves back an hour; 2:00 AM becomes 1:00 AM. If you attempt to synchronize between 1:00 AM and 2:00 AM, the timestamp of the synchronization is ambiguous: is it the first 1:15 AM or the second 1:15 AM?

To resolve this problem you can shut down for an hour when the time changes in the autumn, or you can put your consolidated database server on coordinated universal time (UTC) time.

Related Information

Synchronization Techniques

1.1.5 MobiLink Application Development Options

MobiLink provides a variety of ways to develop an application. You can use these methods alone or in combination.

Create Synchronization Model Wizard

The wizard walks you through the development of your application. You start with a central database that has schema, and you can create remote databases and the scripts needed for synchronization. The wizard can also create shadow tables on your consolidated database to handle things like download deletes. When the wizard completes, you can further customize the model. There is a *Deploy Synchronization Model Wizard* that creates databases and tables, updates the MobiLink system tables, and creates scripts that run MobiLink utilities.

Once you have deployed a MobiLink model, if you have further customizations to it you can still make changes using one of the methods described below.

SQL Central

The MobiLink 17 plug-in for SQL Central enables you to update all the elements of your MobiLink application.

System procedures

When you set up a central database to operate as a consolidated database, system objects are created that are used by MobiLink synchronization. These include MobiLink system tables, where the server side of your MobiLink application is largely stored. They also include system procedures and utilities that you can use to insert MobiLink scripts into your MobiLink system tables, register remote users, and so on.

Direct manipulation of MobiLink system tables

Advanced users may want to add, delete, and update data in the MobiLink system tables directly. Doing so requires an advanced understanding of how MobiLink works.

Related Information

MobiLink Plug-in for SQL Central [page 27]
Synchronization Model Tasks [page 41]
MobiLink Server System Procedures
MobiLink Utilities
MobiLink Server System Tables

1.1.6 Options for Writing Server-side Synchronization Logic

MobiLink synchronization scripts can be written in SQL, or they can be written in Java (using the MobiLink server API for Java) or in .NET (using the MobiLink server API for .NET).

SQL synchronization logic is usually best when synchronizing to a supported consolidated database.

Java and .NET are useful if you are synchronizing against something other than a supported consolidated database. They may also be useful if your design is restricted by the limitations of the SQL language or by the capabilities of your database management system, or if you simply want portability across different RDBMS types.

Java and .NET synchronization logic can function just as SQL logic functions. The MobiLink server can make calls to Java or .NET methods on the occurrence of MobiLink events just as it can access SQL scripts on the occurrence of MobiLink events. When you are working in Java or .NET, you can use the events to do some extra processing, but when you are processing scripts for events that directly handle upload or download rows, your implementation must return a SQL string. With the exception of the two events used in direct row handling, uploads and downloads are not directly accessible from Java or .NET synchronization logic: MobiLink executes the string returned by Java or .NET as SQL.

Direct row handling, which uses the events handle_UploadData and handle_DownloadData to synchronize against a data source, **does** directly manipulate the upload and download rows.

The following are some scenarios where you might want to consider writing scripts in Java or .NET:

Direct row handling

With Java and .NET synchronization logic, you can use MobiLink to access data from data sources other than your consolidated database, such as application servers, web servers, and files.

Authentication

A user authentication procedure can be written in Java or .NET so that MobiLink authentication integrates with your corporate security policies.

Stored procedures

If your RDBMS lacks the ability to use user-defined stored procedures, you can create a method in Java or .NET.

External calls

If your program calls for contacting an external server midway through a synchronization event, you can use Java or .NET synchronization logic to perform actions triggered by synchronization events. Java and .NET synchronization logic can be shared across multiple connections.

Variables

If your database lacks the ability to handle variables, you can create a variable in Java or .NET that persists throughout your connection or synchronization. (Alternatively, with SQL scripts you can use user-defined named parameters, which work with all consolidated database types.

MobiLink Server APIs

Java and .NET synchronization logic are available via the MobiLink server APIs. The MobiLink server APIs are sets of classes and interfaces for MobiLink synchronization.

The MobiLink server API for Java offers you:

- Access to the existing ODBC connection to the consolidated database as a JDBC connection.
- Access to alternate data sources using interfaces such as JDBC, web services, and JNI.
- The ability to create new JDBC connections to the consolidated database to make database changes outside the current synchronization connection. For example, you can use this for error logging or auditing, even if the synchronization connection does a rollback.

- For synchronizing with the consolidated database, the ability to write and debug Java code before it is executed by the MobiLink server. SQL development environments for many database management systems are relatively primitive compared to those available for Java applications.
- Both SQL row handling and direct row handling.
- The full richness of the Java language and its large body of existing code and libraries.

The MobiLink server API for .NET offers you:

- Access to the existing ODBC connection to the consolidated database using Sap classes that call ODBC from .NET.
- Access to alternate data sources using interfaces such as ADO.NET, web services, and OLE DB.
- For synchronizing with the consolidated database, the ability to write and debug .NET code before it is executed by the MobiLink server. SQL development environments for many database management systems are relatively primitive compared to those available for .NET applications.
- Both SQL row handling and direct row handling.
- Code that runs inside the .NET Common Language Runtime (CLR) and allows access to all .NET libraries, including both SQL row handling and direct row handling.

Related Information

Synchronization Scripts
Synchronization Techniques
Synchronization Script Writing in Java
Synchronization Scripts in Microsoft .NET
Direct Row Handling
User-defined Named Parameters
MobiLink Server .NET API Reference
MobiLink Server Java API Reference

1.1.7 The Synchronization Process

A **synchronization** is a process of data exchange between MobiLink clients and a central data source. During this process, the client must establish and maintain a session with the MobiLink server. If successful, the session leaves the remote and consolidated databases in a mutually consistent state.

The client normally initiates the synchronization process. It begins by establishing a connection to the MobiLink server.

The Upload and the Download

To upload rows, MobiLink clients prepare and send an **upload** that contains a list of all the rows that have been updated, inserted, or deleted on the remote database since the last synchronization. Similarly, to download rows, the MobiLink server prepares and sends a **download** that contains a list of inserts, updates, and deletes.

Upload

By default, the MobiLink client automatically keeps track of which rows in the remote database have been inserted, updated, or deleted since the last successful synchronization. Once the connection is established, the MobiLink client uploads a list of all these changes to the MobiLink server.

The upload consists of a set of new and old row values for rows modified in the remote database. (Updates have new and old row values; deletes have old values; and inserts have new values.) If a row has been updated or deleted, the old values are those that were present immediately following the last successful synchronization. If a row has been inserted or updated, the new values are the current row values. No intermediate values are sent, even if the row was modified several times before arriving at its current state.

The MobiLink server receives the upload and executes upload scripts that you define. By default it applies all the changes in a single transaction. When it has finished, the MobiLink server commits the transaction.

Download

The MobiLink server compiles a list of rows to insert, update, or delete on the MobiLink client, using synchronization logic that you create. It downloads these rows to the MobiLink client. To compile this list, the MobiLink server opens a new transaction on the consolidated database.

The MobiLink client receives the download. It takes the arrival of the download as confirmation that the consolidated database has successfully applied all uploaded changes. It ensures that these changes are not sent to the consolidated database again.

Next, the MobiLink client automatically processes the download, deleting old rows, inserting new rows, and updating rows that have changed. It applies all these changes in a single transaction in the remote database. When finished, it commits the transaction.

During synchronization, there are few distinct exchanges of information. The client builds and uploads the entire upload. In response, the MobiLink server builds and downloads the entire download. It is important to limit the verbosity of the protocol when communication is slower and has higher latency, such as when using telephone lines or public wireless networks.

i Note

MobiLink operates using the ODBC isolation level SQL_TXN_READ_COMMITTED as the default isolation level for the consolidated database. If the RDBMS used for the consolidated database supports snapshot isolation, and if snapshot is enabled for the database, then by default MobiLink uses snapshot isolation for downloads.

In this section:

How the Upload Is Processed [page 24]
: When the MobiLink server receives an upload from a MobiLink client, the entire upload is stored until the synchronization is complete.

Referential Integrity and Synchronization [page 25]
: All MobiLink clients, with the exception of UltraLite Java edition, enforce referential integrity when they incorporate the download into the remote database.

Related Information

Overview of MobiLink Events
Events During Upload
Events During Download
MobiLink Isolation Levels

1.1.7.1 MobiLink Events

When a MobiLink client initiates a synchronization, several synchronization events occur. When these events occur, MobiLink looks for a script to match the event. The script contains instructions detailing what you want done.

If you have defined a script for the event and put it in a MobiLink system table, it is invoked.

MobiLink scripts

Whenever an event occurs, the MobiLink server executes the associated script if you have created one. If no script exists, the next event in the sequence occurs.

> i Note
>
> When you use the *Create Synchronization Model Wizard* to create your MobiLink application, all the required MobiLink scripts are created for you. However, you can customize the default scripts, including creating new scripts.

The following are the typical upload scripts for tables. The first event, upload_insert, triggers the running of the upload_insert script, which inserts any changes in the emp_id and emp_name columns into the emp table. The upload_delete and upload_update scripts perform similar functions for delete and update actions on the emp table.

Event	Example script contents
upload_insert	`INSERT INTO emp (emp_id,emp_name) VALUES ({ml r.emp_id}, {ml r.emp_name})`
upload_delete	`DELETE FROM emp WHERE emp_id = {ml r.emp_id}`
upload_update	`UPDATE emp SET emp_name = {ml r.emp_name} WHERE emp_id = {ml r.emp_id}`

The download script uses a cursor. The following is an example of a download_cursor script:

```
SELECT order_id, cust_id
FROM ULOrder
WHERE last_modified >= {ml s.last_table_download}
AND emp_name = {ml r.emp_id}
```

Scripts Can be Written in SQL, Java, or .NET

You can write scripts using the native SQL dialect of your consolidated database, or using Java or .NET synchronization logic. Java and .NET synchronization logic allow you to write code, invoked by the MobiLink server, to connect to a database, manipulate variables, directly manipulate uploaded row operations, or add row operations to the download. There is a MobiLink server API for Java and a MobiLink server API for .NET that provide classes and methods to suit the needs of synchronization.

Storing Scripts

SQL scripts are stored in MobiLink system tables in the consolidated database. For scripts written with the MobiLink server APIs, you store the fully qualified method name as the script. You can add scripts to a consolidated database in several ways:

- When you use the *Create Synchronization Model Wizard*, scripts are stored in the MobiLink system tables when you deploy your project.
- You can manually add scripts to the system tables by using stored procedures that are installed when you set up a consolidated database.
- You can manually add scripts to the system tables using SQL Central.

Related Information

Synchronization Scripts
Synchronization Events
Script Additions and Deletions
Options for Writing Server-side Synchronization Logic [page 16]
MobiLink Consolidated Databases

1.1.7.2 Transactions in the Synchronization Process

The MobiLink server incorporates changes uploaded from each MobiLink client into the consolidated database in one transaction. It commits these changes once it has completed inserting new rows, deleting old rows, making updates, and resolving any conflicts.

> ⚠ Caution
>
> There should be no implicit or explicit commit or rollback in your SQL synchronization scripts or the procedures or triggers that are called from your SQL synchronization scripts. COMMIT or ROLLBACK statements within SQL scripts alter the transactional nature of the synchronization steps. If you use them, MobiLink cannot guarantee the integrity of your data in the event of a failure.

Tracking Downloaded Information

MobiLink uses a last download timestamp, stored in the remote database, to help simplify how downloads are created.

The primary role of the download transaction is to select rows in the consolidated database. If the download fails, the remote database uploads the same last download timestamp over again, and no data is lost.

Begin and End Transactions

The MobiLink client processes information in the download in one transaction. Rows are inserted, updated, and deleted to bring the remote database up to date with the consolidated data.

The MobiLink server uses two other transactions, one at the beginning of synchronization and one at the end. These transactions allow you to record information regarding each synchronization and its duration. So, you can record statistics about attempted synchronizations, successful synchronizations, and the duration of synchronizations. Since data is committed at various points in the process, these transactions also let you commit data that can be useful when analyzing failed synchronization attempts.

Related Information

Last Download Times in Scripts
Overview of MobiLink Events
Events During Upload
Events During Download

1.1.7.3 How Synchronization Failure Is Handled

MobiLink synchronization is fault tolerant. For example, if a communication link fails during synchronization, both the remote database and the consolidated database are left in a consistent state.

On the client, failure is indicated by a return code.

Synchronization failure is handled differently depending on when it happens. The following cases are handled in different ways:

Failure during upload

If the failure occurs while building or applying the upload, the remote database is left in exactly the same state as at the start of synchronization. At the server, any part of the upload that has been applied is rolled back.

Failure between upload and download

If the failure occurs once the upload is complete, but before the MobiLink client receives the download, the client cannot be certain whether the uploaded changes were successfully applied to the consolidated database. The upload might be fully applied and committed, or the failure may have occurred before the server applied the entire upload. The MobiLink server automatically rolls back incomplete transactions in the consolidated database.

The MobiLink client maintains a record of all uploaded changes. The next time the client synchronizes, it requests the state of the previous upload before building the new upload. If the previous upload was not committed, the new upload contains all changes from the previous upload.

Failure during download

When a failure occurs on the remote device while applying the download, any part of the download that has been applied is rolled back and the remote database is left in the same state as before the download.

If you are using non-blocking download acknowledgement, the download transaction has already been committed, but neither the nonblocking_download_ack script nor the publication_nonblocking_download_ack script is invoked.

If you are not using download acknowledgement, there is no server-side consequence of a download failure.

> **i Note**
>
> The MobiLink restartable download feature has functionality that can assist with download failure recovery, and prevent retransmission of the entire download. This functionality has separate implementations for SQL Anywhere and UltraLite remote databases.

No data is lost when a failure occurs. The MobiLink server and the MobiLink client manage this for you. Neither you nor the user need to worry about maintaining consistent data in their application.

Related Information

-ds mlsrv17 Option

1.1.7.4 How the Upload Is Processed

When the MobiLink server receives an upload from a MobiLink client, the entire upload is stored until the synchronization is complete.

This is done for the following reasons:

Filtering download rows

The most common technique for determining which rows to download is to download rows that have been modified since the most recent download. When synchronizing, the upload precedes the download. Any rows that are inserted or updated during the upload are rows that have been modified since the previous download.

It would be difficult to write a download_cursor script that omits from the download rows that were sent as part of the upload. For this reason, the MobiLink server automatically removes these rows from the download.

Processing inserts and updates

By default, tables in the upload are applied to the consolidated database in an order that avoids referential integrity violations. The tables in the upload are ordered based on foreign key relationships. For example, if table A and table C both have foreign keys that reference a primary key column in B, then inserts and updates for table B rows are uploaded first.

Processing deletes after inserts and updates

Deletes are applied to the consolidated database after all inserts and updates are applied. When deletes are applied, tables are processed in the opposite order from the way they appear in the upload. When a row being deleted references a row in another table that is also being deleted, this order of operations ensures that the referencing row is deleted before the referenced row is deleted.

Deadlock

When an upload is being applied to the consolidated database, it may encounter deadlock due to concurrency with other transactions. These transactions might be upload transactions from other MobiLink server database connections, or transactions from other applications using the consolidated database. When an upload transaction is deadlocked, it is rolled back and the MobiLink server automatically starts applying the upload again, from the beginning.

i Note

Performance tip

It is important to write your synchronization scripts to avoid contention as much as possible. Contention has a significant impact on performance when multiple users are synchronizing simultaneously.

1.1.7.5 Referential Integrity and Synchronization

All MobiLink clients, with the exception of UltraLite Java edition, enforce referential integrity when they incorporate the download into the remote database.

Rather than failing the download transaction, by default the MobiLink client automatically deletes all rows that violate referential integrity.

This feature has the following benefits:

- Protection from mistakes in your synchronization scripts. Given the flexibility of the scripts, it is possible to accidentally download rows that would break the integrity of the remote database. The MobiLink client automatically maintains referential integrity without requiring intervention.
- You can use this referential integrity mechanism to delete information from a remote database efficiently. By only sending a delete to a parent record, the MobiLink client removes all the child records automatically for you. This can greatly reduce the amount of traffic MobiLink must send to the remote database.

MobiLink clients provide notification if they have to explicitly delete rows to maintain referential integrity, as follows:

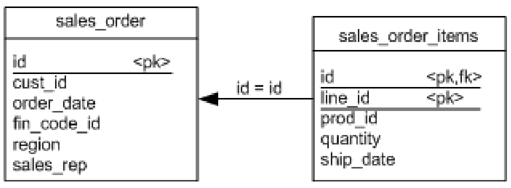
- For SQL Anywhere clients, dbmlsync writes an entry in the log. There are also dbmlsync event hooks that you can use.
- For UltraLite clients, the warning SQLE_ROW_DELETED_TO_MAINTAIN_REFERENTIAL_INTEGRITY is raised. This warning takes a parameter which is the table name. To maintain referential integrity, the warning is raised on every row that is deleted. Your application can ignore the warnings if you want synchronization to proceed. To explicitly handle the warnings, you can use the error callback function to trap them and, for example, count the number of rows that are deleted.
  If you want synchronization to fail when the warning is raised, you must implement a synchronization observer and then signal the observer (perhaps through a global variable) from the error callback function. In this case, synchronization fails on the next call to the observer.

Referential Integrity Checked at the End of the Transaction

The MobiLink client incorporates changes from the download in a single transaction. To offer more flexibility, referential integrity checking occurs at the end of this transaction. Because checking is delayed, the database may temporarily pass through states where referential integrity is violated. This is because rows that violate referential integrity are automatically removed before the download is committed.

Example

Suppose that an UltraLite sales application contains the following two tables. One table contains sales orders. Another table contains items that were sold in each order. They have the following relationship:



If you use the download_delete_cursor for the sales_order table to delete an order, the default referential integrity mechanism automatically deletes all rows in the sales_order_items table that point to the deleted sales order.

This arrangement has the following advantages:

- You do not require a sales_order_items table script because rows from this table are deleted automatically.
- The efficiency of synchronization is improved. You need not download rows to delete from the sales_order_items table. If each sales order contains many items, the performance improves because the download is now smaller. This technique is particularly valuable when using slow communication methods.

Changing the Default Behavior

For SQL Anywhere clients, you can use the sp_hook_dbmlsync_download_ri_violation client event hook to handle the referential integrity violation. Dbmlsync also writes an entry to its log.

Related Information

sp_hook_dbmlsync_download_log_ri_violation
sp_hook_dbmlsync_download_ri_violation

1.1.8 MobiLink Security Considerations

There are several aspects to securing data throughout a widely distributed system such as a MobiLink installation.

Protecting data in the consolidated database

Data in the consolidated database can be protected using the database user authentication system and other security features.

For more information, see your database documentation.

Protecting data in the remote databases

If you are using SQL Anywhere remote databases, the data can be protected using SQL Anywhere security features. By default, these features are designed to prevent unauthorized access through client/server communications, but the default is not fully secure and may not prevent a serious attempt to extract information directly from the database file.

Files on the client are protected by the security features of the client operating system.

Protecting data during synchronization

Communication from MobiLink clients to MobiLink servers can be protected by the MobiLink transport layer security features. Communication from MobiLink servers to the consolidated database can be secured in a database-specific way.

Protecting the synchronization system from unauthorized users

MobiLink synchronization can be secured by a password-based user authentication system. This mechanism prevents unauthorized users from synchronizing data.

Protecting data from users with unnecessary privileges

Through the granting and revoking of roles and privileges, you can ensure that users have the correct authority to perform synchronizations, without having more authority than is desirable.

Related Information

Data Security
Database Security
Transport Layer Security
MobiLink Client/Server Communications Encryption
Digital Certificates
MobiLink Users in a Synchronization System
Authentication Parameters
Security Considerations with Role-based Access Control and Synchronization
-x mlsrv17 Option
-fips mlsrv17 Option

1.2 MobiLink Plug-in for SQL Central

The MobiLink plug-in for SQL Central was redesigned in version 12. In previous versions, the plug-in had two modes: Model mode and Admin mode. The MobiLink functionality was split between these two modes, so you needed to be aware of which mode you were in at any given time.

In version 12 and later, these modes no longer exist.

The top level functions available through the *Folders* pane of the MobiLink plug-in are:

- Working with MobiLink projects.
- Working with consolidated databases.

- Working with MobiLink server command lines.
- Working with remote schema names.
- Working with groups.
- Working with remote tasks.
- Working with synchronization models.

Remote schema name, group and remote tasks are all part of the central administration of remote databases feature.

To start working with MobiLink in SQL Central, you must first define a MobiLink project.

A **MobiLink project** is a framework that organizes the synchronization models, consolidated databases, and remote tasks that are related to a mobile application.

A MobiLink project is a named collection of the following:

- A list of synchronization models
- A list of designed but undeployed remote tasks
- A list of connections to consolidated databases
- A list of user-defined groups of MobiLink users

In this section:

Creating a MobiLink Project [page 29]
Create a MobiLink project before working with MobiLink in SQL Central.

Adding a Consolidated Database [page 30]
Add one or more consolidated databases to a MobiLink project in SQL Central.

Synchronization Models [page 37]
A **synchronization model** is a tool that makes it easy for you to create MobiLink applications. A synchronization model is a file that is created by either the *Create Project Wizard* or the *Create Synchronization Model Wizard* in SQL Central.

Related Information

MobiLink Agents in SQL Central
MobiLink Agents in SQL Central
Remote Tasks
Manage Remote Databases
Creating a MobiLink Server Command Line [page 33]

1.2.1 Creating a MobiLink Project

Create a MobiLink project before working with MobiLink in SQL Central.

Context

A sample MobiLink project is provided in *%SQLANYSAMP17%*`\MobiLink\CustDB\project.mlp`.

Procedure

1. In SQL Central, click ▶ *Tools* ▶ *MobiLink 17* ▶ *New Project* ◀.
2. In the *What do you want to name the new project* field, type a name for the project.
3. In the *Where do you want to save the new project* field, type the location of the project folder or click *Browse* to select the folder for your project file.
4. Click *Next*.
5. On the *Specify a Consolidated Database* page, perform the following tasks:
    a. In the *Database display name* field, type the display name you want to use for the consolidated database. This is the name that is listed in the consolidated database list in your project.
    b. In the *Connection string* field, type the database connection parameters to use to connect to the consolidated database or click *Edit* to go open a window to connect to an ODBC data source.
    c. Select *Remember the password* to save the password used to connect to the database.

        Selecting this option causes the password to be saved in obfuscated form in the project file.
    d. Click *Next*.
6. On the *New Remote Database Schema* page, select the consolidated tables and columns that you want to appear in the remote database. Click *Next*.
7. Select *Add a remote schema name to the project* to identify a group of remote databases that have the same schema or you can add one later. Remote schema names are only useful if you are creating remote tasks.

    If you are adding a remote schema name, specify the following:

    What do you want to name the new remote schema name

    Type the name you want to use to identify the group of remote databases that share the same schema. It might be a good idea to include a version number.

    Click *Next*.
8. On the *Specify A Remote Database Type* page, choose either *SQL Anywhere* or *UltraLite* for the type of remote database you want to use. You can change this setting later from the *Properties* page for the project.
9. Click *Finish* to save the new project.
10. If you are asked to install the MobiLink system setup, click *Yes* and then *OK*.

Results

The MobiLink project is created, a connection to the specified database is established, and a synchronization model is created.

Next Steps

You can now edit or deploy the synchronization model or work with other objects in the MobiLink project, such as remote tasks.

Related Information

Central Administration Concepts

1.2.2 Adding a Consolidated Database

Add one or more consolidated databases to a MobiLink project in SQL Central.

Prerequisites

There must be a MobiLink project defined.

Context

A remote task must have at least one consolidated database assigned to it before it can be deployed.

Procedure

1. Select the MobiLink project.
2. Double-click the project name and click *File* > *New* > *Consolidated Database*.
3. Enter the required database connection parameters and click *Next*.
4. In the *Display name* field, type the name you want to use for this database in your project. The default display name is the ODBC data source name. To provide a description of the database, type it in the *Description* field.

5. Select *Remember the password* to save the password used to connect to the database.

   Selecting this option causes the password to be saved in obfuscated form in the project file.
6. Click *Finish* to add the consolidated database to the project.
7. If you are asked to install the MobiLink system setup, click *Yes* and then *OK*.

Results

The consolidated database is added to the MobiLink project.

In this section:

MobiLink System Setup [page 31]
: You must add objects such as tables, columns, and triggers that are required for synchronization before you can use a database as a MobiLink consolidated database. You add these objects by running a setup script against the database.

Creating a MobiLink Server Command Line [page 33]
: Use the *MobiLink Server Command Line Properties* window in SQL Central to save MobiLink server command lines to include in MobiLink projects.

Adding an LDAP Server [page 34]
: Add a trusted LDAP server using SQL Central. The MobiLink server uses the LDAP server to authenticate users.

Adding a User Authentication Policy [page 35]
: Create a user authentication policy in SQL Central to authenticate a MobiLink user to an LDAP server.

Assigning a User Authentication Policy to Multiple Users [page 36]
: Assign a user authentication policy in SQL Central to multiple users.

Related Information

Central Administration Concepts

1.2.2.1 MobiLink System Setup

You must add objects such as tables, columns, and triggers that are required for synchronization before you can use a database as a MobiLink consolidated database. You add these objects by running a setup script against the database.

There is a separate setup script for each supported RDBMS. These scripts are all located in the *%SQLANY17%* `\MobiLink\setup` folder. You can verify exactly what the script does by opening it in a text editor.

When you add a consolidated database to your MobiLink project, it checks the MobiLink system setup. If it is missing, you are prompted to install the MobiLink system setup. You may also be prompted to upgrade if the check found an older version. You are also prompted to install the MobiLink system setup when you deploy a synchronization model, if you have not already done so.

In this section:

Related Information

Consolidated Database Setup
Synchronization Model Deployment [page 58]

1.2.2.1.1 Checking MobiLink System Setup

Check for the objects required for synchronization using SQL Central.

Prerequisites

There must be a MobiLink project defined and the project must contain at least one consolidated database.

Procedure

1. Double-click the MobiLink project.
2. Double-click *Consolidated Databases* and select the consolidate database you want to check and click *Check MobiLink System Setup*.
3. If setup is not already installed or is not up to date, click *Yes* to install it or update it, then click *OK*.

Results

1.2.2.1.2 Removing MobiLink System Setup

Using SQL Central, remove the objects required for synchronization that were installed by MobiLink system setup.

Prerequisites

There must be a MobiLink project defined with a consolidated database that has MobiLink system setup installed.

Procedure

1. Double-click the MobiLink project.
2. Double-click *Consolidated Databases* and select the consolidate database you want to check and click *Remove MobiLink System Setup*.
3. You are warned that all synchronization models will be removed from the database, the synchronization state will be lost, and the database will be removed from the project. Click *Yes*.

Results

MobiLink system setup is removed from the database and the database is removed from the MobiLink project.

1.2.2.2 Creating a MobiLink Server Command Line

Use the *MobiLink Server Command Line Properties* window in SQL Central to save MobiLink server command lines to include in MobiLink projects.

Prerequisites

There must be a MobiLink project defined.

Procedure

1. Double-click the MobiLink project name.
2. Double-click *MobiLink Server Command Lines*.
3. Click *File* > *New* > *MobiLink Server Command Line*.
4. Type a meaningful name for the command line.
5. Choose a consolidated database from the *Consolidated Database* dropdown list.
6. Click *Add* to specify network options that determine how the MobiLink server listens for synchronization requests. If no options are specified, then the MobiLink server listens for TCP/IP connections on port 2439.
7. Choose a verbosity level from the *Verbosity* dropdown list. Choosing the *Custom* option opens a windows where you can pick from the available verbosity options.
8. Set other desired options on the *Advanced* page. If an option you require is not in the list, you can type it in the *Other options* field.
9. Review the command line displayed in the *Command line* field. Edit the above options if necessary, or click *OK* to save the command line.

Results

The command line is saved with the specified options.

1.2.2.3 Adding an LDAP Server

Add a trusted LDAP server using SQL Central. The MobiLink server uses the LDAP server to authenticate users.

Prerequisites

There must be a MobiLink project defined with at least one consolidated database.

Procedure

1. Double-click the MobiLink project name, and the consolidated database you want to work with.
2. Double-click *LDAP Servers*.
3. Click *File* > *New* > *LDAP Server*.
4. Follow the steps in the *Create LDAP Server Wizard*.
5. Click *Finish*.

Results

The LDAP server is created and displayed in the *LDAP Servers* tab.

Next Steps

You can set up user authentication policies to define how MobiLink users are authenticated using LDAP servers.

Related Information

Adding a User Authentication Policy [page 35]

1.2.2.4 Adding a User Authentication Policy

Create a user authentication policy in SQL Central to authenticate a MobiLink user to an LDAP server.

Prerequisites

There must be at least one LDAP server defined in the consolidated database.

Procedure

1. Double-click the MobiLink project name, and the consolidated database you want to work with.
2. Double-click *User Authentication Policies*.
3. Click *File* > *New* > *User Authentication Policy*.
4. Follow the steps in the *Create User Authentication Policy Wizard*.
5. Click *Finish*.

Results

The user authentication policy is created and displayed on the *User Authentication Policies* tab.

Next Steps

You can create new MobiLink users or modify existing MobiLink users to use the authentication policy.

1.2.2.5 Assigning a User Authentication Policy to Multiple Users

Assign a user authentication policy in SQL Central to multiple users.

Prerequisites

There must be at least one LDAP server and one or more users defined in the consolidated database.

Procedure

1. Double-click the MobiLink project name, and the consolidated database you want to work with.
2. Select *Users*.
3. In the *Details* pane on the right, select the users you want to assign the authentication policy to.
4. Right-click and select *Set Authentication Policy*. Select the authentication policy from the list of currently defined authentication policies.

Results

The user authentication policy is assigned to the selected users.

Next Steps

You can create new MobiLink users or modify existing MobiLink users to use the authentication policy.

1.2.3 Synchronization Models

A **synchronization model** is a tool that makes it easy for you to create MobiLink applications. A synchronization model is a file that is created by either the *Create Project Wizard* or the *Create Synchronization Model Wizard* in SQL Central.

After a synchronization model has been created, you can continue to customize the model. No changes are made to your consolidated database or remote database until you deploy the model. Your model is stored in a model file with the extension `.mlsm`, and a reference to that file is stored in your MobiLink project file.

When your model is complete, you use the *Deploy Synchronization Model Wizard* to deploy it. The *Deploy Synchronization Model Wizard* creates script files to run the MobiLink server and client using the deployment options you chose. You can choose to make changes to your existing databases when you deploy or you can choose to have the wizard create SQL script files that you run later. Files that are created for the remote database can be used in remote tasks.

After you deploy, you can continue to customize the synchronization model or databases and then redeploy. If necessary, you can modify the deployed synchronization system out of SQL Central, using the techniques that are described throughout the MobiLink documentation.

In this section:

1.2.3.1 Setting up a MobiLink Application with the *Create Synchronization Model Wizard*

Use the *Create Synchronization Model Wizard* in SQL Central to set up synchronization logic for a MobiLink application.

Prerequisites

There must be a MobiLink project defined.

Procedure

1. Double-click the MobiLink project. name, and then double-click *Synchronization Models*.
2. Click *File* > *New* > *Synchronization Model* to start the *Create Synchronization Model Wizard*.
3. On the *Welcome* page, choose a name for your synchronization model. Your model is stored as a `.mlsm` file in the project directory. Click *Next*.
4. On the *Primary Key Requirements* page, confirm your system meets the primary key requirements then select the three checkboxes and click *Next*.
5. On the *Consolidated Database Schema* page, select the consolidated database to be used for obtaining your consolidated database schema, and then click *Next*.

   If you chose an Oracle database, you may be prompted to choose a subset of owners because loading the schema for all owners can take a long time.
6. You can create your remote database schema based on the consolidated database schema or an existing remote database. The existing remote database can be SQL Anywhere or UltraLite.
7. Follow the remaining instructions in the *Create Synchronization Model Wizard*. Default recommendations based on best practices are used where possible.
8. Click *Finish*.

Results

When you click *Finish*, the synchronization model is displayed in the right pane of SQL Central, where you can view, edit or deploy the model.

Next Steps

Once you have created a synchronization model, do one of the following:

- Test the synchronization model using the *Test* window.
- Edit the mappings, events and authentication information.
- Use the *Deploy Synchronization Model Wizard* to deploy the completed model.

In this section:

Removing a Synchronization Model [page 39]
    Use the MobiLink plug-in in SQL Central to remove synchronization models from a consolidated database if they were added by using the MobiLink plug-in.

Remote Schemas [page 40]
    A synchronization model contains schema for a remote database. This schema can be obtained from an existing remote database or from the consolidated database.

Related Information

Unique Primary Keys
Synchronization Model Tasks [page 41]
Synchronization Model Deployment [page 58]
Table and Column Mappings [page 42]

1.2.3.1.1 Removing a Synchronization Model

Use the MobiLink plug-in in SQL Central to remove synchronization models from a consolidated database if they were added by using the MobiLink plug-in.

Prerequisites

You must have a synchronization model that was added by using the MobiLink plug-in.

Context

This feature is not available for DB2 or Microsoft Azure databases.

> **Note**
>
> Support for IBM DB2 consolidated databases is deprecated.

Procedure

1. Double-click the project that contains the synchronization model you want to remove.
2. Double-click *Synchronization Models*, right-click the synchronization model you want to remove, and choose *Remove From Consolidated Database*.
3. If the synchronization model is in more than one consolidated database, choose the consolidated database that you want the synchronization model removed from.
4. Click *OK*.

Results

The synchronization model is removed from the selected consolidated database.

1.2.3.1.2 Remote Schemas

A synchronization model contains schema for a remote database. This schema can be obtained from an existing remote database or from the consolidated database.

Use an existing remote database in the following cases:

- If you already have a remote database, especially if its schema is not a subset of the consolidated database schema.
- If your consolidated and remote columns must have different types.
- If your remote tables must have different owners from the tables on the consolidated database. For new SQL Anywhere remote schemas created from the consolidated database, the owner of the remote tables is the same as the owner of the corresponding consolidated database tables. If you want a different owner, use an existing SQL Anywhere remote database with table ownership that you set up.

> i Note
>
> You can manually change an existing database schema and then run the *Update Schema Wizard* to update the synchronization model in your MobiLink project.

When you deploy your model, you have three options for your remote database, regardless of how you created the remote schema in the model. Your deploy-time options for the remote database are:

Create a New Remote Database

Deployment can create a new remote database using the remote schema from the synchronization model. The database is created with default options.

Update an Existing Remote Database That Has No User Tables

You can choose to create or recreate the user tables to be synchronized during deployment. This option is useful when you want to use non-default database creation options, such as a specific collation.

For SQL Anywhere databases, there are options that cannot be set after the database is created.

For UltraLite databases, database properties cannot be changed after the database is created.

Update an Existing Remote Database That Has a Schema Matching the Schema in the Model

This option is useful when you have an existing remote database that you want to synchronize. When you deploy directly to an existing remote database, you can choose to not (re)create the tables to be synchronized. Existing tables and their contents are not changed.

For a SQL Anywhere remote database, tables have the same owners as the original database. UltraLite database tables do not have owners.

Related Information

Synchronization Model Deployment [page 58]
Updating Schemas [page 57]
Initialization Utility (dbinit)

1.2.3.2 Synchronization Model Tasks

You can perform several tasks with your synchronization model after creating it with the *Create Synchronization Model Wizard*. Changes are saved to the synchronization model file only. They are not saved to your consolidated or remote databases until the synchronization model is deployed.

You can modify synchronization models outside of SQL Central but you cannot reverse-engineer changes back into the model. For example, you can add or modify MobiLink scripts by using system procedures.

In this section:

Table and Column Mappings [page 42]
Table mappings indicate which tables should be synchronized, how tables should be synchronized, and how the synchronized data should be mapped between the consolidated and remote databases.

Modifying the Download Strategy [page 47]
Change the download type in the *Download Strategy* tab of the *Details* pane. The download type of a table mapping can be timestamp, snapshot, or custom.

Modifying How Deletes Are Recorded [page 49]
Use the *Download Deletes* tab to control whether deletions on the consolidated database are downloaded to remotes, and how the information about those deletes is stored in the consolidated database.

Download Subsets [page 50]
Each MobiLink remote database can receive a subset of the data in the consolidated database. You can customize the download subset for each table.

Modifying the Download Subset [page 51]
Customize the download subset for each table. Each MobiLink remote database can synchronize a subset of the data in the consolidated database.

Download Delete Subset Modification [page 53]

If you are using a download subset to synchronize a subset of the data on the consolidated database, then by default the download delete subset is set to *Downloaded*, which makes it exactly the same as the download subset. You can choose to change it to *All* or *Custom*.

Related Information

MobiLink Server System Procedures

1.2.3.2.1 Table and Column Mappings

Table mappings indicate which tables should be synchronized, how tables should be synchronized, and how the synchronized data should be mapped between the consolidated and remote databases.

Upload-only, download-only, and non-synchronized tables or columns

By default, MobiLink does a complete, bi-directional synchronization. You can change each table to be upload-only or download-only. You can also choose to not synchronize a table, which removes its table mapping.

In a synchronization model, you can only specify tables as download-only; you cannot create download-only publications.

In this section:

Related Information

Download-only Publications

1.2.3.2.1.1 Changing the Table Mapping Direction

Change the table mapping direction from the *Mappings* tab in SQL Central to indicate how the table should be synchronized or that it should not be synchronized.

Prerequisites

You must have a synchronization model.

Procedure

1. Double-click the MobiLink project.
2. Double-click *Synchronization Models* and double-click your synchronization model name.
3. Click the *Mappings* tab.
4. In the *Table Mappings* pane, select a consolidated table.
5. In the *Direction* dropdown list, select one of the following:

    Not synchronized

    Choosing this option is the same as deleting the table mapping.

    Bi-directional

    Database operations are synchronized from the remote database and the consolidated database, and vice versa.

    Download to remote only

    Changes are synchronized from the consolidated database to the remote database only.

    Upload to consolidated only

    Changes are synchronized from the remote database to the consolidated database only.

Results

The table mappings are updated.

1.2.3.2.1.2 Removing a Table Mapping

Removing a table mapping stops the table from being synchronized. Table mappings can be removed from the *Mappings* tab in SQL Central

Prerequisites

You must have a synchronization model.

Procedure

1. Double-click the MobiLink project name.
2. Double-click *Synchronization Models* and double-click your synchronization model name.
3. Click the *Mappings* tab.
4. In the *Table Mappings* pane, select a table mapping.
5. In the *Direction* dropdown list, click *Not synchronized*.

Results

The mapping is deleted the next time you save the synchronization model.

1.2.3.2.1.3 Changing Table Mappings

Check and customize the table and column mappings. If your model is based on an existing remote database, the table and column mappings represent a best guess.

Prerequisites

You must have a synchronization model.

Procedure

1. Double-click the MobiLink project name.

2. Double-click *Synchronization Models* and double-click your synchronization model name.
3. Click the *Mappings* tab.
4. In the *Table Mappings* pane, click the remote table for the mapping you want to change.
5. Click the ellipsis (three dots) button next to the remote table name and select a different table from the list of unsynchronized remote tables.
    - You can only choose remote tables that are not already mapped to consolidated tables.
    - To add tables to your remote schema, see the *Updating schemas* topic.

Results

The table mappings are updated.

Related Information

Updating Schemas [page 57]

1.2.3.2.1.4 Adding an Unmapped Table to a Mapping

Add tables in the consolidated database that have not been mapped for synchronization to the synchronization model.

Prerequisites

You must have a synchronization model.

Procedure

1. Double-click the MobiLink project name.
2. Double-click *Synchronization Models* and double-click your synchronization model name.
3. Click the *Mappings* tab.
4. Click *File* > *New* > *Table Mappings* to open the *Create New Table Mappings* window where you choose consolidated tables to add.
5. Select one or more of the consolidated tables to add them to the remote schema. The new remote tables have the same names and the same sets of columns as the corresponding consolidated tables, and the mappings between the remote and consolidated tables are created automatically.

The following options are available:

Add the chosen tables to the remote schema if they don't already exist

If you don't want to make changes to the remote schema, disable this option.

Let me choose the columns for the new remote tables

Chose this option to add the tables but not all columns to the remote schema. Enabling this option lets you chose columns for each selected table.

Hide tables named like synchronization model shadow tables (since those should not be synchronized)

By default consolidated database tables with names like synchronization model shadow table names are not shown, since such shadow tables should not be synchronized.

Results

The new tables are added to the mapping for the synchronization model.

1.2.3.2.1.5 Changing a Column Mapping for a Table Mapping

Check and customize table and column mappings as required. If your model is based on an existing remote database, the table and column mappings represent a best guess.

Prerequisites

You must have a synchronization model.

Context

You can map a column in a synchronized consolidated table to a remote table column, a value determined when synchronizing, or exclude the column from synchronization. When mapping to a value, you can use the MobiLink user name, the remote database ID, or a SQL expression (which can include MobiLink named parameters). When you map a primary key column to a value and the table mapping is bi-directional, you must prevent duplicate primary keys when downloading to remote databases.

Procedure

In the *Table Mappings* tab, right-click the column mapping you want to change, and select one of the following options from the *Download Subset* context menu:

- *None*
- *User*
- *Remote*
- *Custom*
- An unmapped remote column

To synchronize the consolidated column with a remote column, select the unmapped remote column from the bottom group of the menu. Only unmapped remote columns are listed.

To exclude the consolidated column from synchronization, click *None*. The Direction icon shows that the consolidated column is not synchronized.

To map the consolidated column to a value, you can choose *User*, *Remote*, or use *Custom* to enter a SQL expression that is evaluated when the remote table's upload_insert, upload_update, and upload_delete synchronization scripts are executed during the synchronization. The Direction icon shows that the value is only uploaded; the consolidated column is not downloaded to the remote database.

Results

The column mappings are updated.

1.2.3.2.2 Modifying the Download Strategy

Change the download type in the *Download Strategy* tab of the *Details* pane. The download type of a table mapping can be timestamp, snapshot, or custom.

Prerequisites

You must have a synchronization model.

Context

The download types are as follows:

Timestamp-based download

Choose this option to use timestamp-based download as the default. Only rows that have been changed since the last synchronization are downloaded.

Snapshot download

Choose this option to use snapshot download as the default. All rows are downloaded on each synchronization even if they have not been changed since the last synchronization.

Custom download logic

Choose this option to write your own download_cursor and download_delete_cursor scripts instead of having them generated for you.

Procedure

1. Double-click the MobiLink project name.
2. Double-click *Synchronization Models* and double-click your synchronization model name.
3. Open the *Mappings* tab.
4. In the *Table Mappings* pane, select a table mapping.
5. In the *Details* pane, select the *Download Strategy* tab.
6. From the *Strategy* dropdown list, select *Timestamp*, *Snapshot*, or *Custom*.
7. If you chose *Custom*, click the *Events* tab in the upper pane and then type in your download_cursor script and download_delete_cursor scripts.
8. If you chose *Timestamp*, perform the following tasks:
   a. Enter a column name in the *Timestamp column name* field.
   b. Choose one of the following:
      1. *Update the timestamp column using*
         - *Default column values*
         - *Triggers*
      2. *Store timestamp column in a shadow table*

Results

The download type is modified.

Related Information

Snapshot Synchronization
Snapshot Synchronization
Synchronization Scripts
download_cursor Scripts
download_delete_cursor Scripts
Implementing Timestamp-based Downloads

1.2.3.2.3 Modifying How Deletes Are Recorded

Use the *Download Deletes* tab to control whether deletions on the consolidated database are downloaded to remotes, and how the information about those deletes is stored in the consolidated database.

Prerequisites

You must have a synchronization model and the download type for the table mapping must be set to *Timestamp*.

Context

If you are using snapshot download, all rows in the remote database are deleted before the snapshot is downloaded so there is no need to track delete operations. If you are using timestamp-based download, you can decide how you want deletes on the consolidated database to be recorded for downloading to the remote database.

To delete rows from remote databases when they are deleted from the consolidated database, you must keep a record of each deleted row. You can record this information with shadow tables or by using logical deletes.

Procedure

1. Double-click the MobiLink project name.
2. Double-click *Synchronization Models* and then select your synchronization model name.
3. Click the *Mappings* tab.
4. In the *Table Mappings* pane, select a table mapping.
5. In the *Details* pane, open the *Download Deletes* tab.
6. Select the *When rows are deleted in the consolidated database, delete them from the remote databases* checkbox to download deletes from the consolidated database. Clear the checkbox if you do not want to download deletes from the consolidated database.

   To record deletions, you can choose to use a shadow table or logical deletes.

Results

The method of handling deletes is updated.

Related Information

Deletes
download_delete_cursor Scripts
download_cursor Table Event

1.2.3.2.4 Download Subsets

Each MobiLink remote database can receive a subset of the data in the consolidated database. You can customize the download subset for each table.

The download subset options are:

User

Choose this option to partition data by MobiLink user name, which downloads different data to different registered MobiLink users.

To use this option, each row must contain a column that contains a MobiLink user name. You choose your MobiLink user names when you deploy, so you can choose names that match existing values on your consolidated database. (The column that you use for MobiLink user names must be of a type that can hold the values you are using for the user name.) If the MobiLink user names are in a different table from the one you are subsetting, you must join to that table.

Remote

Choose this option to partition data by remote ID, which downloads different data to different remote databases.

To use this option, each row must contain a column that holds a remote ID. Remote IDs are created as GUIDs by default, but you can set the remote IDs to match existing values on your consolidated database. (The column you use for remote IDs must be of a type that can hold the values you are using for the remote IDs.) If the remote IDs are in a different table from the one you are subsetting, you must join to that table.

> i Note
>
> It is usually better to partition by user or by authentication parameter than by remote ID, because the remote ID can change if the remote computer is reset or replaced.

Custom

Choose this option to use a SQL expression that determines which rows are downloaded. Each synchronization only downloads rows where your SQL expression is true. This SQL expression is added to the WHERE clause of the generated download_cursor script. You can use MobiLink named parameters in the expression. You can also refer to other tables. If you refer to other tables, you must list the other tables in the field above the expression, and include the join condition in your expression.

Related Information

MobiLink Users in a Synchronization System

Remote IDs
Partitioned Rows Among Remote Databases
Remote IDs and MobiLink User Names in Scripts
Last Download Times in Scripts
download_cursor Scripts

1.2.3.2.5 Modifying the Download Subset

Customize the download subset for each table. Each MobiLink remote database can synchronize a subset of the data in the consolidated database.

Prerequisites

You must have a synchronization model and the table mapping must not have a download type set to *Custom*.

Procedure

1. Double-click the MobiLink project name.
2. Double-click *Synchronization Models* and select your synchronization model name.
3. Click the *Mappings* tab.
4. In the *Table Mappings* pane, select a remote table.
5. In the *Details* pane, open the *Download Subset* tab.
6. Choose one of the following download subsets from the *Download Subset* dropdown list: *None*, *User*, *Remote*, or *Custom*.
7. If you chose *User* or *Remote*, identify where the column containing the user name and remote ID is located.

   If the column is in the consolidated table being synchronized, select *Table* `table name` and then select the column containing the user name or remote ID from the *Column name* dropdown list.

   If the column is in a different table, select *Another table that can be joined with table* `table name`. Select the table containing the column from the *Other table* dropdown list. Select the column containing the user name or remote ID in the *Column containing MobiLink user* dropdown list. Use the join condition to define a join condition for joining the synchronizing table to the shadow table.
8. If you chose *Custom*, there are two text boxes where you add information to construct a download_cursor script. You do not have to write a complete download_cursor. Just add extra information to identify the join and other restrictions for the download subset.
   - In the first text box (*SQL expression*), enter a SQL expression to be added to the generated WHERE clause that specifies the download subset condition and join condition. You can use MobiLink named parameters, including authentication parameters, in the expression. By default, the same expression and joined tables are used for the download delete subset. If you are using a shadow table to track deletes and want to use the same expression, avoid using the base table name in the expression. If that is not possible, use a custom download delete subset.

- In the second text box (*Additional tables*), enter the table name(s) if your download_cursor requires a join to other tables. If the join requires multiple tables, separate them with commas.

Results

The download subset is modified.

Example

User example

For example, the ULOrder table in CustDB can be shared between users. By default, orders are assigned to the employee who created them, but there are times when another employee needs to see orders created by someone else. For example, a manager may need to see all the orders created by employees in their department. The CustDB database has a provision for this case via the ULEmpCust table. It allows you to assign customers to employees. They download all orders for that employee customer relationship.

View the download_cursor script for ULOrder without download subsetting. Select the ULEmpCust table in the *Mapping* tab. Choose *Timestamp* for the *Download Strategy* and *None* for the *Download Subset*. Right-click the table and click *Go To Events*. The download_cursor for the table looks like this:

```
SELECT "DBA"."ULOrder"."order_id",
    "DBA"."ULOrder"."cust_id",
    "DBA"."ULOrder"."prod_id",
    "DBA"."ULOrder"."emp_id",
    "DBA"."ULOrder"."disc",
    "DBA"."ULOrder"."quant",
    "DBA"."ULOrder"."notes",
    "DBA"."ULOrder"."status"
FROM "DBA"."ULOrder"
WHERE "DBA"."ULOrder"."last_modified" >= {ml s.last_table_download}
```

Now go back to the *Mappings* tab. In the *Download Subset* tab of the *Details* pane, change the *Download Subset* dropdown list for ULOrder to *User*. Select *Another table that can be joined with table DBA.ULOrder*. For the other table, select ULEmpCust. For the column containing MobiLink user, select emp_id. For the join condition, select DBA.ULOrder.cust_id=DBA.ULEmpCust.cust_id.

Right-click the table in the top pane and click *Go To Events*. The download_cursor for the table now looks like this (the new lines are shown in bold):

```
SELECT "DBA"."ULOrder"."order_id",
    "DBA"."ULOrder"."cust_id",
    "DBA"."ULOrder"."prod_id",
    "DBA"."ULOrder"."emp_id",
    "DBA"."ULOrder"."disc",
    "DBA"."ULOrder"."quant",
    "DBA"."ULOrder"."notes",
    "DBA"."ULOrder"."status"
FROM "DBA"."ULOrder", "DBA"."ULEmpCust"
WHERE "DBA"."ULOrder"."last_modified" >= {ml s.last_table_download}
AND "DBA"."ULOrder"."cust_id" = "DBA"."ULEmpCust"."cust_id"
AND "DBA"."ULEmpCust"."cust_id" = {ml s.username}
```

Custom example

Assume you want to subset the download of a table called `Customer` by MobiLink user and you also want to only download rows where active=1. The MobiLink user names do not exist in the table you are subsetting, so you must create a join to a table called SalesRep, which contains the MobiLink user names.

In the *Mappings* tab, select the mapping for the `Customer` table. Open the *Download Strategy* tab in the *Details* pane. Set the download type to *Timestamp*. Open the *Download Subset* tab in the *Details* pane. Select *Custom* for the *Download Subset* dropdown list. In the first box (*SQL expression*), type:

```
SalesRep.ml_username = {ml s.username}
   AND Customer.active = 1
   AND Customer.cust_id = SalesRep.cust_id
```

In the second box (*Additional tables*), type:

```
SalesRep
```

Both tables have a cust_id column, so references to those columns have to be prefixed with the table name in the expression. If you use a shadow table for tracking deletes to be downloaded, you must use *None* or *Custom* in the *Download Delete Subset* column for the Customer table mapping, since the shadow table is called Customer_del instead of Customer.

Right-click the table in the top pane and click *Go To Events*. The download_cursor for the table now looks like this:

```
SELECT "DBA"."Customer"."cust_id",
 "DBA"."Customer"."cust_name"
FROM "DBA"."Customer", SalesRep
WHERE "DBA"."Customer"."last_modified" >= {ml s.last_table_download}
 AND SalesRep.ml_username = {ml s.username}
 AND Customer.active = 1
 AND Customer.cust_id = SalesRep.cust_id
```

The final line of the WHERE clause creates a key join of Customer to SalesRep.

1.2.3.2.6 Download Delete Subset Modification

If you are using a download subset to synchronize a subset of the data on the consolidated database, then by default the download delete subset is set to *Downloaded*, which makes it exactly the same as the download subset. You can choose to change it to *All* or *Custom*.

If the download subset is *Custom* and you are using a shadow table to track deletes, any column values used by your download subset logic must be copied to the shadow table when a row is deleted. Primary key column values are automatically copied to the shadow table.

Except for specifying the extra columns to be added to the shadow table, defining a custom download subset is identical to setting up a custom download delete subset.

Related Information

Modifying the Download Subset [page 51]

1.2.3.2.7 Conflict Detection and Resolution

When a row is updated on both the remote and consolidated databases, a conflict occurs the next time that the databases are synchronized.

You have the following options for detecting conflicts:

Row-based conflict detection

A conflict is detected if the row has been updated by both the remote and consolidated databases since the last synchronization.

This option defines an upload_fetch script and upload_update script.

Column-based conflict detection

A conflict is detected if the same column has been updated for the row in both the remote and consolidated databases.

This option defines an upload_fetch_column_conflict script.

If a table has BLOBs and you choose column-based conflict detection, row-based conflict detection is used.

You have the following options for resolving conflicts:

Consolidated

First in wins: uploaded updates that conflict are discarded.

Remote

Last in wins: uploaded updates are always applied.

Timestamp

The newest update wins. To use this option, you must create and maintain a TIMESTAMP column for the table. This TIMESTAMP column should record the last time that a row was changed. The column should exist on both the consolidated and remote databases and not be the same column used for timestamp-based downloads. To work, your remote and consolidated databases must use the same time zone (preferably UTC) and their clocks must be synchronized.

Custom

You write your own resolve_conflict scripts. You do this on the *Events* tab.

In this section:

Modifying Conflict Detection and Resolution [page 55]
Use SQL Central to customize the settings for conflict detection and resolution.

Related Information

Conflict Detection with upload_fetch or upload_fetch_column_conflict Scripts
Conflict Detection with upload_fetch or upload_fetch_column_conflict Scripts
Conflict Handling Overview
Conflict Resolution with resolve_conflict Scripts

1.2.3.2.7.1 Modifying Conflict Detection and Resolution

Use SQL Central to customize the settings for conflict detection and resolution.

Prerequisites

You must have a synchronization model.

Procedure

1. Double-click the MobiLink project name.
2. Double click your synchronization model name.
3. Click the *Mappings* tab.
4. In the *Table Mappings* pane, select a table mapping.
5. In the *Details* pane, open the *Conflict Handling* tab.
6. Under *Choose a strategy for detecting conflicts*, choose *Row-based* or *Column-based*.
7. Choose *First in wins*, *Last in wins*, *Custom* from the *Choose a strategy for resolving conflicts when they are detected* dropdown list.
8. If you chose *Custom* conflict resolution, open the *Events* tab and write a resolve_conflict script for the table.

Results

The conflict detection and resolution settings are updated.

1.2.3.2.8 Script Modification in a Synchronization Model

You modify scripts in a synchronization model using the *Events* tab.

The *Events* tab allows you to perform the following tasks:

- View and modify the scripts generated based on the current table mappings.
- Create new scripts.

The top of the *Events* tab indicates the group that the selected script belongs to. All scripts for a single table are grouped together. The top of the *Events* tab also indicates the name of the selected script and whether it was generated by the MobiLink plug-in, whether it was user-defined, or whether a generated script was overridden. It also indicates whether the synchronization logic is written in SQL, .NET, or Java.

The script is fully under your control when you add a script to override a generated script; it does not change automatically when you change a related setting. For example, if you change a download_delete_cursor for a model and then clear the *Del* column in the *Table Mappings* pane under the *Mappings* tab, your customized download_delete_cursor is not affected.

You can use options in the *File* menu to restore generated scripts you have overridden, restore scripts that you have set to be ignored, or to remove new scripts you have added. Select the script(s) you want to restore or remove and click *File* to view your options.

In this section:

1.2.3.2.8.1 Locating a Script for a Particular Table

Use SQL Central to locate the scripts defined for a particular table.

Prerequisites

You must have a synchronization model.

Procedure

1. Double-click the MobiLink project.
2. Double-click your synchronization model name.
3. Open the *Events* tab.
4. In the *Group* dropdown, select the table whose events you want to see
5. In the *Event* field, choose the script name that you want to locate. Existing scripts are highlighted in bold.

Results

The cursor moves to the selected script.

Next Steps

Make any desired changes to the selected script.

1.2.3.3 Updating Schemas

Use the *Update Schema Wizard* to update the consolidated and remote database schemas in your synchronization model.

Prerequisites

You must have a synchronization model.

Context

The *Update Schema Wizard* is most useful after you have deployed your model and:

- You made a change to the remote database schema that needs to be included in the model.
- You made a change to the consolidated database schema that needs to be included in the model.
  For example, you need to run Update Schema before redeploying a model that created timestamp-based download for one or more tables. The previous deployment changed the schema of the consolidated database by adding a TIMESTAMP column or shadow table, so the schema needs to be updated.

Procedure

1. Double-click the MobiLink project name.
2. Double-click *Synchronization Models*.
3. Double-click your synchronization model name and click *File* > *Update schema*.
4. Choose one of the following options:

   The Consolidated Database Schema

   The consolidated schema in the model is updated. The remote schema in the model is unchanged.

The Remote Database Schema

The remote schema in the model is updated. The consolidated schema in the model is unchanged.

Both The Consolidated And Remote Database Schemas

Both the consolidated and remote schemas are updated in the model to match the schemas of the existing databases.

5. Follow the instructions in the *Update Schema Wizard*.

   When you click *Finish*, no changes are made outside the model until you deploy the synchronization model; the consolidated database does not change and the remote database is not created or changed until that time.
6. Map the new remote tables in the *Mappings* tab.

Results

The schema is updated.

Related Information

Table and Column Mappings [page 42]

1.2.3.4 Synchronization Model Deployment

You deploy synchronization models with the *Deploy Synchronization Model Wizard*.

The following items can be deployed:

- Changes to the consolidated database.
- SQL Anywhere or UltraLite remote databases (you can choose to create a database, or add tables to an existing empty database, or use an existing database that already has your remote tables).
- Batch files to deploy the model (the generated batch files have variable declarations at the beginning that you can edit before running the batch files).
- Batch files to run the MobiLink server and the MobiLink client.

In the MobiLink 17 plug-in, schemas are compared when you deploy a synchronization model. Only database objects (for example, tables, indexes, and so on) that are different from the objects being deployed are modified. You are warned if deploying the new objects will break the existing synchronization system and you are given the chance to abort the operation. This functionality is not available for IBM DB2 LUW consolidated databases.

> i Note
>
> Support for IBM DB2 consolidated databases is deprecated.

Redeploying updates to the same script version automatically removes schema no longer required by the new script version. For example, if a last_modified column is added to a table to support downloads but you change to upload only synchronization, then the last_modified column is dropped during deployment.

You can also use the ml_model_drop system procedure to remove a synchronization model and its schema from the consolidated database. This system procedure drops the synchronization scripts and any schema that was created when the synchronization model was deployed, including shadow tables, tracking columns, triggers, and indexes. Schema that is shared with another script_version is not deleted. Only schema installed using the MobiLink 17 plug-in can be dropped. This functionality is not available for IBM DB2 LUW consolidated databases.

Deploying to the consolidated database

The *Deploy Synchronization Model Wizard* provides two options for deploying to the consolidated database:

- Apply your synchronization model directly to your consolidated database by populating MobiLink system tables and creating all required shadow tables, columns, triggers, and stored procedures.
- Create a UTF-8 encoded SQL file that contains all the same changes and a batch file to run the SQL file against your consolidated database. You can inspect this file, alter it, and run it anytime. The effect is identical to applying the changes directly.

In this section:

Synchronization model files are located in the MobiLink project directory. A synchronization file has the file extension `.mlsm`. When you deploy a synchronization model, a directory starting with your model name and ending with `_deploy` is created.

Running the Batch Files [page 67]
Run the batch files that are created by the *Deploy Synchronization Model Wizard* from the command line to synchronize your synchronization model.

Related Information

ml_model_drop System Procedure

1.2.3.4.1 Testing a Synchronization Model Before Deployment

Use the *Test* window to test a synchronization model before you deploy it.

Prerequisites

You must have a synchronization model defined in your MobiLink project.

Context

Changes are made to the consolidated database when you use the test feature.

Procedure

1. Double-click the MobiLink project name
2. Double-click *Synchronization Models* and double-click your synchronization model name.
3. In the *Deployment* pane, click *Test*. You are warned that testing the synchronization model causes changes to the consolidated database and modifies data in tables. Click *OK*.
4. Click *Synchronize*.

Results

The synchronization model is tested and you can see whether the synchronization was successful.

Next Steps

You can do the following tasks when your test completes:

- Review the information about the *Data* tab to compare the data in the consolidated and remote databases.
- Review the information about the *Client Log* tab for information about failures during the test. This is only available for SQL Anywhere clients.
- Review the information about the *MobiLink Log* tab for information about failures during the test.
- Modify the remote and/or consolidated database before synchronizing again. There are several options for modifying the databases:
  - Use the *Data* tab to directly modify data in the tables being synchronized.
  - Use the *Actions* menu to open the dbisql utility on either database.

1.2.3.4.2 Configuring Synchronization Model Testing

Use the *Test Window Configuration* window to configure some of the options used for testing the synchronization model, enabling more complete testing of your synchronization logic and creating an environment closer to that of your production environment.

Prerequisites

You must have a synchronization model defined in your MobiLink project.

Context

Changes are made to the consolidated database when you use the test feature.

Procedure

1. Double-click the MobiLink project name.
2. Double-click *Synchronization Models* and select your synchronization model name.
3. In the *Deployment* pane, click *Test*. You are warned that testing the synchronization model causes changes to the consolidated database and modifies data in tables. Click *OK*.

4. Click *Actions* > *Configure*.
5. Fill in the following fields:

   MobiLink User

   Type the MobiLink user name you want the MobiLink client to use when connecting to the MobiLink server.

   MobiLink Password

   Type the password for the MobiLink user name you specified.

   Authentication Parameters

   Type the authentication parameters, using a comma-separated list. For example, `p1`,`p2`,`p3`. These values are sent from the MobiLink client to the MobiLink server and can be accessed by the server-side synchronization scripts.

   MobiLink Server Command Line

   Choose a MobiLink server command line to use when starting the MobiLink server for testing. To review the options specified on the MobiLink server command line, click *View*. The *MobiLink Server Command Line Properties* window appears.

   Client Network Protocol

   Choose the network communication protocol for the MobiLink client to use when connecting to the MobiLink server. Only protocols defined by the MobiLink server command line can be chosen.

   Client Network Options

   Choose network protocol options for the MobiLink client to use when connecting to the MobiLink server. Edit the options directly or click *Edit* to edit the options from the *Client Network Options* page.

   When you select a client network protocol, default options are generated in this field based on the MobiLink server command line. Always review the default options to ensure that they are adequate for synchronization.

   Synchronization Type

   Choose one of the following synchronization types from the dropdown list.

      Bidirectional

      Database operations are synchronized from the remote database to the consolidated database, and vice versa.

      Download only

      Changes are synchronized from the consolidated database to the remote database only.

      Upload only

      Changes are synchronized from the remote database to the consolidated database only.

   Always use these settings when testing this synchronization model with this consolidated database

   Check this option to use this configuration every time you test this synchronization model. The specified options are used until the settings are changed.
6. Click *OK*. You are warned that the synchronization model will be redeployed and that data in the remote database will be lost. Click *OK*.

Results

The synchronization model is redeployed in the *Test* window using the new settings.

Next Steps

You can perform the following tasks once the synchronization model has been redeployed:

- Use *Actions* > *Configure* to change the *Test* window settings again.
- Modify the remote and/or consolidated database. There are several options for modifying the databases:
  - Use the *Data* tab to directly modify data in the tables being synchronized.
  - Use the *Actions* menu to open the dbisql utility on either database.
- Run a test synchronization by clicking *Synchronization*.

Related Information

MobiLink Users in a Synchronization System
Authentication Parameters
Authentication Process
Upload-only and Download-only Synchronizations
Testing a Synchronization Model Before Deployment [page 60]
Creating a MobiLink Server Command Line [page 33]
MobiLink Client Network Protocol Options

1.2.3.4.3 Deploying a Consolidated Database from a SQL File

Use the *Deploy Synchronization Model Wizard* to deploy a consolidated database.

Prerequisites

If your deployment creates shadow tables, you must connect to the consolidated database as either the owner of the base tables for which shadow tables are created, or as an administrator.

Procedure

When you ran the *Deploy Synchronization Model Wizard*, if you chose to create a file to run later (on the *Choose How To Prepare Databases For Synchronization* page), you must run the batch file that is located in the `project-name_deploy` subfolder of your model. This file creates all the objects you chose to have created in the consolidated database, including synchronization scripts, shadow tables, and triggers. It can also register MobiLink users in the consolidated database.

To run this file, navigate to the `project-name_deploy` directory and run the file `cons_setup.bat` or `cons_setup.sh`. You must include connection information. For example, run:

```
cons_setup "DSN=my_odbc_datasource;UID=myuserid;PWD=mypassword"
```

For some drivers, the ODBC data source can include the user ID and password so they do not need to be specified.

Results

The consolidated database is deployed.

1.2.3.4.4 Remote Database Deployment

You can choose to use an existing remote database or have the wizard create one for you. The wizard can create remote databases directly or you can have it create a UTF-8 encoded SQL file and a batch file that you run to create or update remote databases.

The wizard creates a remote database (either SQL Anywhere or UltraLite) with default database creation options using the database owner that you specified in the model. Alternatively, you can create a remote database outside the *Deploy Synchronization Model Wizard* with your own custom settings and use the wizard to add the required remote tables, or you can deploy to an existing remote database that already has the remote tables.

1.2.3.4.5 Deploying a Remote Database from a SQL File

Use the *Deploy Synchronization Model Wizard* to deploy a remote database.

Procedure

When you ran the *Deploy Synchronization Model Wizard*, if you chose to create a file to run later (on the *Choose How To Prepare Databases For Synchronization* page), you must run the batch file that was created with the SQL file in the `project-name_deploy` directory. This file creates all the objects you chose to have created in the remote database, including tables, publications, subscriptions, and MobiLink users.

To run this file, navigate to the `project-name_deploy` directory and run the file `remote_setup.bat` or `remote_setup.bat.sh`. For example, run:

```
remote_setup.bat
```

You are prompted for a password if you are using an existing remote database.

Results

The remote database is deployed.

1.2.3.4.6 Batch file Deployment to Run Synchronization Tools

The *Deploy Synchronization Model Wizard* can create batch files that can be used to run synchronization tools.

The following batch files can be created by the *Deploy Synchronization Model Wizard*:

- A batch file to run the MobiLink server with options that you specify.
- For SQL Anywhere remote databases, a batch file to run dbmlsync with options that you specify.
- For UltraLite remote databases, a batch file to run ulsync with options that you specify. Ulsync is used for testing synchronization, so it helps you get started when you don't have a working UltraLite application.

1.2.3.4.7 Synchronization Model Redeployment

You can alter a synchronization model after deploying it by making changes to the synchronization model and then redeploying.

You can also alter your model by using system procedures or other methods. However, you cannot reverse-engineer the changes back into the synchronization model when you alter a deployed model outside of SQL Central. Changes made outside of SQL Central are overwritten when you redeploy the model.

Deployment often causes schema changes, so you may need to update the schema even if you haven't made any other changes. For example, if you deploy a model that adds a TIMESTAMP column to each synchronized table on the consolidated database (which is the default behavior when you create a model), you must update the consolidated schema in the model before redeploying. Likewise, if you add a table to the consolidated database and then want to redeploy, you must update the consolidated schema in the model and then create new remote tables.

Related Information

Updating Schemas [page 57]

1.2.3.4.8 Deployed Synchronization Models

Synchronization model files are located in the MobiLink project directory. A synchronization file has the file extension `.mlsm`. When you deploy a synchronization model, a directory starting with your model name and ending with `_deploy` is created.

Depending on the deployment options you chose, you might have the following files:

File	Description
`cons_setup.bat`	A batch file that runs the SQL file to set up the consolidated database.
`cons_setup.sql`	A SQL file for setting up the consolidated database.
`mlsrv.bat`	A batch file for running the MobiLink server.
`remote_setup.bat`	A batch file that runs the SQL file to set up the remote database.
`remote_setup.sql`	A SQL file for setting up the consolidated database.
`summary.txt`	A file that summarizes the details of the synchronization model.
`sync.bat`	If you deployed a SQL Anywhere remote database, then this is a batch file for synchronizing SQL Anywhere databases with dbmlsync.
`modelname_remote.db`	If you chose to create a new SQL Anywhere remote database, then this is the database file.
`modelname_remote.udb`	If you chose to create a new UltraLite remote database, then this is the database file.
`modelname_ulsync.bat`	If you deployed an UltraLite database, then this is a batch file for testing synchronization with an UltraLite remote database using the ulsync utility.

Related Information

Initiation of Synchronization
GRANT ROLE Statement

1.2.3.4.9 Running the Batch Files

Run the batch files that are created by the *Deploy Synchronization Model Wizard* from the command line to synchronize your synchronization model.

Prerequisites

You must have a synchronization model.

Context

If you have not yet run MobiLink setup scripts on the consolidated database, run them before deploying.

To run many of the batch files, you must include connection information. You may also need to create ODBC data sources before running these batch files.

Procedure

1. When you ran the *Deploy Synchronization Model Wizard*, if you chose to create a file to run later (on the *Choose How To Prepare Databases For Synchronization* page), then you must run the batch file that is located in the `project-name_deploy` subfolder of your model. This file creates all the objects you chose to have created in the consolidated database, including synchronization scripts, shadow tables, and triggers. It can also register MobiLink users in the consolidated database.

   To run this file, navigate to the `project-name_deploy` directory and run the file `cons_setup.bat` or `cons_setup.sh`. You must include connection information on the command line. For example, run:

   ```
   cons_setup.bat "DSN=MY_ODBC_DATASOURCE"
   ```
2. When you ran the *Deploy Synchronization Model Wizard*, if you chose to create a file to run later (on the *Choose How To Prepare Databases For Synchronization* page), then you must run the batch file in the `project-name_deploy` directory. This file creates all the objects you chose to have created in the remote database, including tables, publications, subscriptions, and MobiLink users.

   To run this file, navigate to the `project-name_deploy` directory and run the file `remote_setup.bat` or `remote_setup.sh`. For example, run:

   ```
   remote_setup.bat
   ```

   You are prompted for a password if you are using an existing remote database.
3. Start the MobiLink server by running `mlsrv.bat`. You must include connection information for the consolidated database on the command line. For example, run:

   ```
   mlsrv.bat "DSN=MY_ODBC_DATASOURCE"
   ```

4. Synchronize.

   For a SQL Anywhere remote database:

   - Grant the SYS_RUN_REPLICATION_ROLE system role to a user to run dbmlsync. For example, execute the following in Interactive SQL:

     ```
     GRANT ROLE SYS_RUN_REPLICATION_ROLE
     TO userid
     ```

   - Connect as that user to the database.
   - Start the remote database that is located in the `project-name_deploy` directory. For example, run:

     ```
     dbsrv17 MyModel_remote.db
     ```

   - Start dbmlsync, the SQL Anywhere MobiLink client. Run the file `sync.bat` in the `project-name_deploy` directory. You must include connection information on the command line. For example, run:

     ```
     sync.bat "UID=userid;PWD=password;SERVER=MyModel_remote"
     ```

   For an UltraLite remote database:

   - To test your synchronization, run the file that ends with `_ulsync.bat` in the `remote` directory.
   - Alternatively, run your UltraLite application.

Results

The synchronization model is redeployed.

Related Information

ODBC Data Sources
Consolidated Database Setup

1.2.3.5 Limitations of Synchronization Models

Synchronization models have some limitations.

Changes made outside the model cannot be redeployed

If you deploy a synchronization model and then make changes to it outside the model, those changes are not saved in the model. This practice is fine to use the model as a starting point, deploy, and then make all your changes outside the model. However, to redeploy the model, you are better off making your changes to your MobiLink project so that they are saved and can be redeployed.

Versions

A synchronization model can have only one version.

MobiLink system database

You cannot use a MobiLink system database that is separate from the consolidated database when deploying a synchronization model.

Multiple publications

You cannot create multiple publications. After you have deployed your model, you can add more publications using non-model methods such as the CREATE PUBLICATION statement, but you cannot reverse-engineer these additions back into your model.

Views

It is not possible to select a view when you are selecting consolidated database tables for table mappings.

Computed columns

You cannot upload to computed columns in a consolidated database table. If you deploy a synchronization model with computed columns, then the deployment may have errors creating the trigger used for timestamp-based downloads. You can either exclude the column from synchronization, or configure the table as download-only (and either use snapshot download or edit the generated consolidated SQL file to remove the computed column from the trigger definition).

Copying computed columns causes a syntax error when deploying the new remote schema to create a new remote database. When dealing with computed columns, do one of the following:

- Deploy the synchronization model to an existing remote database.
- Exclude the computed column from the remote schema. To synchronize a consolidated database table that has computed columns, you cannot upload to the table.

The Microsoft SQL Server AdventureWorks sample database contains computed columns. Set the columns to be download-only or exclude the columns from synchronization when using this database to create a model.

Oracle XMLTYPE data type

The MobiLink plug-in for SQL Central provides the following support for the Oracle XMLTYPE.

- When an XMLTYPE column is chosen for a new remote schema or when editing column mappings, the displayed *Data Type* name is *XMLTYPE*.
- XMLTYPE columns are mapped to columns of type XML in new SQL Anywhere remote databases.
- The generated synchronization scripts work when XMLTYPE column values are 4 KB or less. For larger values, you must override the generated scripts in the synchronization model with scripts using the techniques described in the Oracle XMLTYPE data type topic.

Proxy tables

In the remote database, proxy tables cannot be part of a publication.

Deployment Considerations

Spatial columns

Spatial columns are copied, though the spatial subtype and SRID may not be copied if the consolidated RDBMS does not have metadata support for obtaining those, such as the ST_GEOMETRY_COLUMNS view of the SQL/MM standard. Spatial support in UltraLite is limited to one type (ST_GEOMETRY) that only supports point values and column SRID constraints of SRID=0 or SRID=4326, so you may get a warning or error when deploying an incompatible spatial type to a new UltraLite database.

Long object names

The database objects that are created when deploying may have names that are longer than the database supports (because the new object names are created by adding suffixes to the base table names). If this happens, deploy only to file (not directly to a database) and edit the generated SQL file to replace all occurrences of the name that is too long.

New remote schemas

If you create a new remote schema in the *Create Synchronization Model Wizard*, the new remote database columns do not contain indexes of the columns in the consolidated database. Foreign keys and default column values are copied to the new remote database; however, this support relies on database metadata returned by the ODBC driver and syntax or other errors may occur due to driver problems. For example, if a driver reports a default column value in a format that cannot be used to declare such a default in a SQL Anywhere or UltraLite remote database, then errors can occur (including syntax errors when deploying).

UltraLite does not support NCHAR(n), NVARCHAR(n), or LONG NVARCHAR column types. When deploying a synchronization model to a new UltraLite database, such columns in the remote schema are converted to CHAR(4n), VARCHAR(4n), or LONG VARCHAR. If 4n is larger than the maximum length for CHAR and VARCHAR, then the maximum length is used and you get a warning.

You can use an existing remote database to create a synchronization model or to update the remote schema in a model.

Proxy tables

It is possible to synchronize with consolidated database tables that are proxy tables to another database, but you must add the TIMESTAMP column to both the base table and the proxy table if you use a TIMESTAMP column for timestamp-based downloads. The *Deploy Synchronization Model Wizard* cannot add a column to a proxy table or its base, so you must either use an existing column on both the base and proxy, or you must use a shadow table or snapshot download.

Materialized views

If you are using timestamp-based downloads and have chosen to add a TIMESTAMP column to consolidated tables, you must disable any materialized views that depend on the tables before deploying. Otherwise, you may get errors when trying to alter the tables. For SQL Anywhere consolidated databases, use the sa_dependent_views system procedure to find out if a table has dependent materialized views.

Other Considerations

Creating remote databases based on an Oracle consolidated database

When you are using an Oracle consolidated database as the basis for your SQL Anywhere or UltraLite remote database, you may want to change DATE columns in the consolidated database to TIMESTAMP. Otherwise, sub-second information is lost on upload.

Related Information

MobiLink System Database
Oracle XMLTYPE Data Type

Script Versions
Publications
Oracle Data Mapping
sa_dependent_views System Procedure

1.3 CustDB Sample for MobiLink

CustDB is a sales-status application. The CustDB sample is a valuable resource for the MobiLink developer. It provides you with examples of how to implement many of the techniques you need to develop MobiLink applications.

The application has been designed to illustrate several common synchronization techniques. You can study the sample application as you read to better understand synchronization techniques.

A version of CustDB is supplied for each supported operating system and for each supported database type.

A MobiLink project that uses CustDB consolidated database is available in the *%SQLANYSAMP17%*`\MobiLink\CustDB\project.mlp` directory. You can open this project in SQL Central to work with CustDB projects and view database scripts.

CustDB Scenario

A consolidated database is located at the head office. The following data is stored in the consolidated database:

- The MobiLink system tables that hold the synchronization metadata, including the synchronization scripts that implement synchronization logic.
- The CustDB data, including all customer, product, and order information, stored in the rows of base tables.

There are two types of remote databases, mobile managers and sales representatives.

Each mobile sales representative's database contains all products but only those orders assigned to that sales representative, while a mobile manager's database contains all products and orders for a specific set of customers.

Synchronization Design

The synchronization design in the CustDB sample application uses the following features:

Complete table downloads

All rows and columns of the ULProduct table are shared in their entirety with the remote databases.

Column subsets

All rows, but not all columns, of the ULCustomer table are shared with the remote databases.

Row subsets

Different remote users get different sets of rows (row subsets) from the ULOrder table.

Timestamp-based synchronization

This is a way of identifying changes that were made to the consolidated database since the last time a device synchronized. The ULCustomer and ULOrder tables are synchronized using a method based on timestamps.

Snapshot synchronization

This is a simple method of synchronization that downloads all rows in every synchronization. The ULProduct table is synchronized in this way.

Primary key pools to maintain unique primary keys

It is essential to ensure that primary key values are unique across a complete MobiLink installation. The primary key pool method used in this application is one way of ensuring unique primary keys.

In this section:

Related Information

The CustDB Sample Database Application

CustDB Consolidated Database Setup [page 73]

Partitioned Rows Among Remote Databases
Snapshot Synchronization
Primary Key Pools
Unique Primary Keys
Tutorial: Building the UltraLite CustDB Sample Application
Tutorial: Building the UltraLite CustDB Sample Application
Implementing Timestamp-based Downloads

1.3.1 CustDB Files

The pieces that make up the code for the CustDB sample application and database include the following.

- The sample SQL scripts, located in the *%SQLANYSAMP17%*`\MobiLink\CustDB`.
- The application code, located in *%SQLANYSAMP17%*`\UltraLite\CustDB`.
- Platform-specific user interface code, located in subdirectories of *%SQLANYSAMP17%*`\UltraLite\CustDB` named for each operating system.

In this section:

1.3.1.1 CustDB Consolidated Database Setup

The CustDB consolidated database can be any MobiLink supported consolidated database.

SQL Anywhere 17 CustDB

A SQL Anywhere consolidated database is provided in *%SQLANYSAMP17%*`\UltraLite\CustDB\custdb.db`. A ODBC data source called SQL Anywhere 17 CustDB is included with your installation.

You can rebuild this database using the file *%SQLANYSAMP17%*`\UltraLite\CustDB\makedbs.cmd`.

To explore the way the CustDB sample is created, you can view the file *%SQLANYSAMP17%*`\MobiLink\CustDB\custdb.sql`.

CustDB for Other RDBMSs

The following SQL scripts are provided in *%SQLANYSAMP17%*`\MobiLink\CustDB` to build the CustDB consolidated database as any one of these supported RDBMSs:

RDBMS	Custdb setup script
Adaptive Server Enterprise	`custase.sql`
Microsoft Azure	`custmss.sql`
Microsoft SQL Server	`custmss.sql`
Oracle	`custora.sql`
DB2 LUW (deprecated)	`custdb2.sql`
MySQL	`custmys.sql`

In this section:

Building CustDB as a Consolidated Database (Adaptive Server Enterprise, MySQL, Oracle, Microsoft SQL Server, and Microsoft Azure) [page 74]
    The following procedure creates a CustDB consolidated database for any of the supported RDBMS.

Building CustDB as a Consolidated Database (DB2 LUW) (Deprecated) [page 76]
    Use this procedure to set up a DB2 LUW consolidated database.

Related Information

Consolidated Database Setup

1.3.1.1.1 Building CustDB as a Consolidated Database (Adaptive Server Enterprise, MySQL, Oracle, Microsoft SQL Server, and Microsoft Azure)

The following procedure creates a CustDB consolidated database for any of the supported RDBMS.

Prerequisites

You must have access to the SQL scripts that are used to build the CustDB consolidated database as one of the supported RDBMSs. The SQL scripts are located in *%SQLANYSAMP17%*`\MobiLink\CustDB`.

Procedure

1. Create a database in your RDBMS.
2. Add the MobiLink system objects by running one of the following SQL scripts, located in the `MobiLink\setup` subdirectory of your SQL Anywhere 17 installation:
    - For an Adaptive Server Enterprise consolidated database, run `syncase.sql`.
    - For a MySQL consolidated database, run `syncmys.sql`.
    - For an Oracle consolidated database, run `syncora.sql`.
    - For a Microsoft SQL Server or Microsoft Azure consolidated database, run `syncmss.sql`.
3. Add sample user tables, stored procedures and MobiLink synchronization scripts to the CustDB database by running one of the following SQL scripts, located in *%SQLANYSAMP17%*`\MobiLink\CustDB`:
    - For an Adaptive Server Enterprise consolidated database, run `custase.sql`.
    - For a MySQL consolidated database, run `custmys.sql`.
    - For an Oracle consolidated database, run `custora.sql`.
    - For a Microsoft SQL Server or Microsoft Azure consolidated database, run `custmss.sql`.
4. Create an ODBC data source called CustDB that references your database on the client computer.
    a. Choose *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *ODBC Data Source Administrator* (32 or 64 bit).
    b. Click *Add*.
    c. Select the appropriate driver from the list.

    Click *Finish*.
    d. Name the ODBC data source CustDB.
    e. Click the *Login* tab. Enter the *User ID* and *Password* for your database.
5. Click *OK* and then click *OK* again.

Results

A CustDB consolidated database is created for the selected RDBMS.

1.3.1.1.2 Building CustDB as a Consolidated Database (DB2 LUW) (Deprecated)

Use this procedure to set up a DB2 LUW consolidated database.

Procedure

1. Create a consolidated database on the DB2 LUW server named **CustDB**.
2. Ensure that the default table space (usually called USERSPACE1) uses 8 KB pages.

   If the default table space does not use 8 KB pages, complete the following steps:

   a. Create a new table space and temporary table space with 8 KB pages.Verify that at least one of your buffer pools has 8 KB pages. If not, create a buffer pool with 8 KB pages.

      For more information, consult your DB2 LUW documentation.
3. Add the MobiLink system objects to the DB2 LUW consolidated database using the file `MobiLink\setup\syncdb2.sql`:

   a. Change the connect command at the top of the file `syncdb2.sql`. Replace `DB2Database` with the name of your database (or its alias). In this example, the database is called CustDB. You can also add your DB2 user name and password as follows:

   ```
   connect to CustDB user userid using password ~
   ```

   b. Verify that at least one of your buffer pools has 8 KB pages. If not,Open a DB2 LUW Command Window on either the server or client computer. Run `syncdb2.sql`

   ```
   db2cmd db2 -c -ec -td~ +s -v -f syncdb2.sql
   ```
4. by typing theAdd data tables, stored procedures and MobiLink synchronization scripts to the CustDB database:

   a. If necessary, change the connect command in custdb2.sql. For example, you could add the user name and password as follows. Replace `userid` and `password` with your user name and password.

   ```
   connect to CustDB user userid using password
   ```

   b. Open a DB2 Command Window on either the server or client computer.
   c. Run custdb2.sql by typing the following command:

   ```
   db2cmd db2 -c -ec -td~ +s -v -f custdb2.sql
   ```

   d. When processing is complete, enter the following command to close the command window:

   ```
   exit
   ```
5. Create an ODBC data source called CustDB that references the DB2 LUW database on the DB2 LUW client.

   a. Start the ODBC Data Source Administrator:

      Choose *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *ODBC Data Source Administrator*.

The ODBC Data Source Administrator appears.

b. On the *User DSN* tab, click *Add*.
c. In the *Create New Data Source* window, select the ODBC driver for your DB2 LUW database. For example, choose IBM DB2 UDB ODBC Driver. Click *Finish*.

   For information about how to configure your ODBC driver, see:

   - Your DB2 LUW documentation
   - Recommended ODBC Drivers For MobiLink

Results

The DB2 LUW consolidated database is set up.

Related Information

IBM DB2 LUW Consolidated Database
Recommended ODBC Drivers For MobiLink

1.3.1.2 UltraLite Remote Database Setup

The following example creates a remote database for CustDB. The CustDB remote database must be an UltraLite database.

The application logic for the remote database is located in *%SQLANYSAMP17%*`\UltraLite\CustDB`. It includes the following files:

C++ API logic

The file `custdbcpp.cpp` contains the C++ API logic.

User-interface features

These features are stored separately, in platform-specific subdirectories of `Samples\UltraLite\CustDB`.

To install the sample application to a remote device that is running UltraLite, do the following:

1. Start the consolidated database.
2. Start the MobiLink server.
3. Install and start the sample application to your client device.
4. Synchronize the sample application.

Example

The following example illustrates how to install the CustDB sample on a Windows desktop running against a DB2 consolidated database.

> **Note**
>
> Support for IBM DB2 as a consolidated database is deprecated.

1. Ensure that the consolidated database is running:
   For a DB2 LUW database, open a DB2 Command Window. Type the following command, where `userid` and `password` are the user ID and password for connecting to the DB2 LUW database:

   ```
   db2 connect to CustDB user userid using password
   ```
2. Start the MobiLink server:
   For a DB2 LUW database, at a command prompt, run the following command:

   ```
   mlsrv17 -c "DSN=SQL Anywhere 17 CustDB;uid=ml_server;pwd=sql" -zp
   ```
3. Start the CustDB sample application:
   1. Click *Start* > *Programs* > *SQL Anywhere 17* > *MobiLink* > *Synchronization Server Sample*.
   2. Enter a value of 50 for the employee ID and click *OK*.
      The application automatically synchronizes and a set of customers, products, and orders are downloaded to the application from the CustDB consolidated database.
4. Synchronize the remote application with the consolidated database.
   From the *File* menu, choose *Synchronize*.
   You only need to complete this step when you have made changes to the database.

1.3.2 Tables in the CustDB Databases

The table definitions for the CustDB database are in platform-specific files in *%SQLANYSAMP17%*`\MobiLink\CustDB`.

Both the consolidated and the remote databases contain the following five tables, although their definitions are slightly different in each location.

ULCustomer

The ULCustomer table contains a list of customers.

In the remote database, ULCustomer has the following columns:

cust_id

A primary key column that holds a unique integer that identifies the customer.

cust_name

A 30-character string containing the name of the customer.

In the consolidated database, ULCustomer has the following additional column:

last_modified

A timestamp containing the last time the row was modified. This column is used for timestamp-based synchronization.

ULProduct

The ULProduct table contains a list of products.

In both the remote and consolidated databases, ULProduct has the following columns:

prod_id

A primary key column that contains a unique integer that identifies the product.

price

An integer identifying the unit price.

prod_name

A 30-character string that contains the name of the product.

ULOrder

The ULOrder table contains a list of orders, including details of the customer who placed the order, the employee who took the order, and the product being ordered.

In the remote database, ULOrder has the following columns:

order_id

A primary key column that holds a unique integer identifying the order.

cust_id

A foreign key column referencing ULCustomer.

prod_id

A foreign key column referencing ULProduct.

emp_id

A foreign key column referencing ULEmployee.

disc

An integer containing the discount applied to the order.

quant

An integer containing the number of products ordered.

notes

A 50-character string containing notes about the order.

status

A 20-character string describing the status of the order.

In the consolidated database, ULOrder has the following additional column:

last_modified

A timestamp containing the last time the row was modified. This column is used for timestamp-based synchronization.

ULOrderIDPool

The ULOrderIDPool table is a primary key pool for ULOrder.

In the remote database, ULOrderIDPool has the following column:

pool_order_id

A primary key column that holds a unique integer identifying the order ID.

In the consolidated database, ULOrderIDPool has the following additional columns:

pool_emp_id

An integer column containing the employee ID of the owner of the remote database to which the order ID has been assigned.

last_modified

A timestamp containing the last time the row was modified.

ULCustomerIDPool

The ULCustomerIDPool table is a primary key pool for ULCustomer.

In the remote database, ULCustomerIDPool has the following column:

pool_cust_id

A primary key column that holds a unique integer identifying the customer ID.

In the consolidated database, ULCustomerIDPool has the following additional columns:

pool_emp_id

An integer column containing the employee ID that is used for a new employee generated at a remote database.

last_modified

A timestamp containing the last time the row was modified.

The following tables are contained in the consolidated database only:

ULIdentifyEmployee_nosync

The ULIdentifyEmployee_nosync table exists both in the consolidated and remote database. It has a single column as follows:

emp_id

This primary key column contains an integer representing an employee ID in the *remote* database.

ULEmployee

The ULEmployee table exists only in the consolidated database. It contains a list of sales employees.

ULEmployee has the following columns:

emp_id

A primary key column that holds a unique integer identifying the employee.

emp_name

A 30-character string containing the name of the employee.

ULEmpCust

The ULEmpCust table controls which customers' orders are downloaded. If the employee needs a new customer's orders, inserting the employee ID and customer ID forces the orders for that customer to be downloaded.

emp_id

A foreign key to ULEmployee.emp_id.

cust_id

A foreign key to ULCustomer.cust_id. The primary key consists of emp_id and cust_id.

action

A character used to determine if an employee record should be deleted from the remote database. If the employee no longer requires a customer's orders, set to D (delete). If the orders are still required, the action should be set to null.

A logical delete must be used in this case so that the consolidated database can identify which rows to remove from the ULOrder table. Once the deletes have been downloaded, all records for that employee with an action of D can also be removed from the consolidated database.

last_modified

A timestamp containing the last time the row was modified. This column is used for timestamp-based synchronization.

ULOldOrder and ULNewOrder

These tables exists only in the consolidated database. They are for conflict resolution and contain the same columns as ULOrder. In SQL Anywhere, Microsoft Azure, and Microsoft SQL Server, these are temporary tables. In Adaptive Server Enterprise, these are normal tables and @@spid. DB2 LUW and Oracle do not have temporary tables, so MobiLink needs to identify which rows belong to the synchronizing user. Since these are base tables, if five users are synchronizing, they might each have a row in these tables at the same time.

> i Note
>
> Support for IBM DB2 consolidated databases is deprecated.

Related Information

SQL Variables
The CustDB Sample Database Application

1.3.3 Users in the CustDB Sample

There are two types of users in the CustDB sample, sales people and mobile managers.

The differences are as follows:

Sales people

User IDs 50, 51, and 52 identify remote databases that are associated with sales people. Sales people can perform the following tasks:

- View lists of customers and products.
- Add new customers.
- Add or delete orders.
- Scroll through the list of outstanding orders.
- Synchronize changes with the consolidated database.

Mobile managers

User ID 53 identifies the remote database associated with the mobile manager. The mobile manager can perform the same tasks as a sales person. In addition, the mobile manager can perform the following task:

- Accept or deny orders.

1.3.4 Synchronization Logic Source Code

You can use SQL Central to inspect the synchronization scripts in the consolidated database.

Script Types and Events

The `custdb.sql` file adds each synchronization script to the consolidated database by calling ml_add_connection_script or ml_add_table_script.

Example

The following lines in `custdb.sql` add a table-level script for the ULProduct table, which is executed during the download_cursor event. The script consists of a single SELECT statement.

```
call ml_server.ml_add_table_script(
'CustDB 17.0',
'ULProduct', 'download_cursor',
'SELECT prod_id, price, prod_name FROM ULProduct' )
go
```

1.3.5 Synchronization of Orders in the CustDB Sample

The business rules for the ULOrder table are as follows.

- Orders are downloaded only if they are not approved or the status is null.
- Orders can be modified at both the consolidated and remote databases.
- Each remote database contains only the orders assigned to an employee.

Downloads

Orders can be inserted, deleted, or updated at the consolidated database. The scripts corresponding to these operations are as follows:

download_cursor

The first parameter in the download_cursor script is the last download timestamp. It is used to ensure that only rows that have been modified on either the remote or the consolidated database since the last synchronization are downloaded. The second parameter is the employee ID. It is used to determine which rows to download.

The download_cursor script for CustDB is as follows:

```
CALL ULOrderDownload( {ml s.last_table_download}, {ml s.username} )
```

The ULOrderDownload procedure for CustDB is as follows:

```
CREATE PROCEDURE ULOrderDownload ( IN LastDownload timestamp, IN EmployeeID
integer )
BEGIN
  SELECT o.order_id, o.cust_id, o.prod_id, o.emp_id, o.disc, o.quant,
o.notes, o.status
    FROM ULOrder o, ULEmpCust ec
    WHERE o.cust_id = ec.cust_id
    AND ec.emp_id = EmployeeID
    AND ( o.last_modified >= LastDownload
    OR ec.last_modified >= LastDownload)
    AND ( o.status IS NULL  OR  o.status != 'Approved' )
    AND ( ec.action IS NULL )
END
```

download_delete_cursor

The download_delete_cursor script for CustDB is as follows:

```
SELECT o.order_id
    FROM ULOrder o, dba.ULEmpCust ec
  WHERE o.cust_id = ec.cust_id
    AND ( ( o.status = ''Approved'' AND o.last_modified >= {ml
s.last_table_download} )
    OR ( ec.action = ''D''  )  )
    AND ec.emp_id = {ml s.username}
```

Uploads

Orders can be inserted, deleted or updated at the remote database. The scripts corresponding to these operations are as follows:

upload_insert

The upload_insert script for CustDB is as follows:

```
INSERT INTO ULOrder ( order_id, cust_id, prod_id, emp_id, disc, quant, notes,
status )
   VALUES( {ml r.order_id, r.cust_id, r.prod_id, r.emp_id, r.disc, r.quant,
r.notes, r.status } )
```

upload_update

The upload_update script for CustDB is as follows:

```
UPDATE ULOrder
  SET cust_id = {ml r.cust_id},
      prod_id = {ml r.prod_id},
      emp_id = {ml r.emp_id},
      disc = {ml r.disc},
      quant = {ml r.quant},
      notes = {ml r.notes},
      status = {ml r.status}
    WHERE order_id = {ml r.order_id}
```

upload_delete

The upload_delete script for CustDB is as follows:

```
DELETE FROM ULOrder WHERE order_id = {ml r.order_id}
```

upload_fetch

The upload_fetch script for CustDB is as follows:

```
SELECT order_id, cust_id, prod_id, emp_id, disc, quant, notes, status
   FROM ULOrder WHERE order_id = {ml r.order_id}
```

upload_old_row_insert

The upload_old_row_insert script for CustDB is as follows:

```
INSERT INTO ULOldOrder ( order_id, cust_id, prod_id, emp_id, disc, quant, 
notes, status )
   VALUES( {ml r.order_id, r.cust_id, r.prod_id, r.emp_id, r.disc, r.quant, 
r.notes, r.status } )
```

upload_new_row_insert

The upload_new_row_insert script for CustDB is as follows:

```
INSERT INTO ULNewOrder ( order_id, cust_id, prod_id, emp_id, disc, quant, 
notes, status )
   VALUES( {ml r.order_id, r.cust_id, r.prod_id, r.emp_id, r.disc, r.quant, 
r.notes, r.status } )
```

Conflict Resolution

resolve_conflict

The resolve_conflict script for CustDB is as follows:

```
CALL ULResolveOrderConflict
```

The ULResolveOrderConflict procedure for CustDB is as follows:

```
CREATE PROCEDURE ULResolveOrderConflict()
BEGIN
  -- approval overrides denial
  IF 'Approved' = (SELECT status FROM ULNewOrder) THEN
    UPDATE ULOrder o
    SET o.status = n.status, o.notes = n.notes
    FROM ULNewOrder n
    WHERE o.order_id = n.order_id;
  END IF;
  DELETE FROM ULOldOrder;
  DELETE FROM ULNewOrder;
END
```

1.3.6 Synchronization of Customers in the CustDB Sample

The business rules governing customers are as follows: customer information can be modified at both the consolidated and remote databases; both the remote and consolidated databases contain a complete listing of customers.

Downloads

Customer information can be inserted or updated at the consolidated database. The script corresponding to these operations is as follows:

download_cursor

The following download_cursor script downloads all customers for whom information has changed since the last time the user downloaded information.

```
SELECT cust_id, cust_name FROM ULCustomer WHERE last_modified >= {ml
s.last_table_download}
```

Uploads

Customer information can be inserted, updated, or deleted at the remote database. The scripts corresponding to these operations are as follows:

upload_insert

The upload_insert script for CustDB is as follows:

```
INSERT INTO ULCustomer( cust_id, cust_name )
   VALUES( {ml r.cust_id, r.cust_name } )
```

upload_update

The upload_update script for CustDB is as follows:

```
UPDATE ULCustomer SET cust_name = {ml r.cust_name}
   WHERE cust_id = {ml r.cust_id}
```

Conflict detection is not performed on this table.

upload_delete

The upload_delete script for CustDB is as follows:

```
DELETE FROM ULCustomer WHERE cust_id = {ml r.cust_id}
```

1.3.7 Synchronization of Products in the CustDB Sample

All rows are downloaded for ULProduct; this is called snapshot synchronization.

The business rules for the ULProduct table are as follows:

- Products can only be modified at the consolidated database.
- Each remote database contains all the products.

Downloads

Product information can be inserted, deleted, or updated at the consolidated database. The script corresponding to these operations is as follows:

download_cursor

The following download_cursor script downloads all the rows and columns of the ULProduct table at each synchronization:

```
SELECT prod_id, price, prod_name FROM ULProduct
```

Related Information

Snapshot Synchronization

1.3.8 Maintenance of the Customer and Order Primary Key Pools

The CustDB sample database uses primary key pools to maintain unique primary keys in the ULCustomer and ULOrder tables. The primary key pools are the ULCustomerIDPool and ULOrderIDPool tables.

In this section:

1.3.8.1 ULCustomerIDPool

The following scripts are defined in the ULCustomerIDPool table.

Downloads

begin_download

The begin_download script for CustDB is as follows:

```
CALL ULCustomerIDPool_maintain( {ml s.username} )
```

The ULCustomerIDPool_maintain procedure for CustDB is as follows:

```
CREATE PROCEDURE ULCustomerIDPool_maintain ( IN syncuser_id INTEGER )
BEGIN
  DECLARE pool_count INTEGER;
  -- Determine how many ids to add to the pool
  SELECT COUNT(*) INTO pool_count
    FROM ULCustomerIDPool
    WHERE pool_emp_id = syncuser_id;
  -- Top up the pool with new ids
  WHILE pool_count < 20 LOOP
    INSERT INTO ULCustomerIDPool ( pool_emp_id )
      VALUES ( syncuser_id );
    SET pool_count = pool_count + 1;
  END LOOP;
END
```

download_cursor

```
SELECT pool_cust_id FROM ULCustomerIDPool
   WHERE last_modified >= {ml s.last_table_download}
     AND pool_emp_id = {ml s.username}
```

Uploads

upload_delete

The upload_delete script for CustDB is as follows:

```
DELETE FROM ULCustomerIDPool
   WHERE pool_cust_id = {ml r.pool_cust_id}
```

1.3.8.2 ULOrderIDPool

The following scripts are defined in the ULOrderIDPool table.

Downloads

begin_download

The begin_download script for CustDB is as follows:

```
CALL ULOrderIDPool_maintain( {ml s.username} )
```

The ULOrderIDPool_maintain procedure for CustDB is as follows:

```
ALTER PROCEDURE ULOrderIDPool_maintain ( IN syncuser_id INTEGER )
BEGIN
  DECLARE pool_count INTEGER;
  -- Determine how many ids to add to the pool
  SELECT COUNT(*) INTO pool_count
    FROM ULOrderIDPool
    WHERE pool_emp_id = syncuser_id;
  -- Top up the pool with new ids
  WHILE pool_count < 20 LOOP
    INSERT INTO ULOrderIDPool ( pool_emp_id )
      VALUES ( syncuser_id );
    SET pool_count = pool_count + 1;
  END LOOP;
END
```

download_cursor

The download_cursor script for CustDB is as follows:

```
SELECT pool_order_id FROM ULOrderIDPool
  WHERE last_modified >= {ml s.last_table_download}
    AND pool_emp_id = {ml s.username}
```

Uploads

upload_delete

The upload_delete script for CustDB is as follows:

```
DELETE FROM ULOrderIDPool
   WHERE pool_order_id = {ml r.pool_order_id}
```

1.3.9 How to Restore the CustDB Database

To restore the sample, run the following command from the *%SQLANYSAMP17%*`\UltraLite\CustDB` directory.

```
makedbs
```

1.4 MobiLink Contact Sample

The Contact sample is a valuable resource for the MobiLink developer. It provides you with an example of how to implement many of the techniques you need to develop MobiLink applications.

The Contact sample application includes a SQL Anywhere consolidated database and two SQL Anywhere remote databases. It illustrates several common synchronization techniques. To get the most benefit, study the sample application as you read.

Although the consolidated database is a SQL Anywhere database, the synchronization scripts consist of SQL statements that should work with minimal changes on other database management systems.

The Contact sample is in *%SQLANYSAMP17%*`\MobiLink\Contact`.

Synchronization Design

The synchronization design in the Contact sample application uses the following features:

Column subsets

A subset of the columns of the Customer, Product, SalesRep, and Contact tables on the consolidated database are shared with the remote databases.

Row subsets

All the columns, but only one of the rows of the SalesRep table on the consolidated database are shared with each remote database.

Timestamp-based synchronization

This is a way of identifying changes that were made to the consolidated database since the last time a device synchronized. The Customer, Contact, and Product tables are synchronized using a method based on timestamps.

Privileges

You must have the following roles and privileges:

- SYS_REPLICATION_ADMIN_ROLE system role

- SYS_RUN_REPLICATION_ROLE system role
- CREATE ANY TRIGGER system privilege

In this section:

Related Information

Partitioned Rows Among Remote Databases
Implementing Timestamp-based Downloads

1.4.1 Contact Sample Setup

A Microsoft Windows batch file called `build.bat` is provided to build the Contact sample databases. On UNIX and Linux systems, the file is `build.sh`.

Context

You may want to examine the contents of the batch file. It performs the following actions:

- Creates ODBC data source definitions for a consolidated database and each of two remote databases.
- Creates a consolidated database named `consol.db` and loads the MobiLink system tables, database schema, some data, synchronization scripts, and MobiLink user names into the database.
- Creates two remote databases, each named `remote.db`, in subdirectories named `remote_1` and `remote_2`. Loads information common to both databases and applies customizations. These customizations include a global database identifier, a MobiLink user name, and subscriptions to two publications.

Procedure

1. At a command prompt, navigate to *%SQLANYSAMP17%*`\MobiLink\Contact`.
2. Run `build.bat` (Microsoft Windows) or `build.sh` (UNIX/Linux).

Results

The Contact sample database is built.

In this section:

1.4.1.1 Running the Contact Sample

The Contact sample includes batch files that perform initial synchronizations and illustrate MobiLink server and dbmlsync command lines.

Prerequisites

Ensure that `build.bat` has been run to create the Contact sample database.

Context

You can examine the contents of the following batch files, located in *%SQLANYSAMP17%*`\MobiLink\Contact`, in a text editor:

- `step1.bat`
- `step2.bat`
- `step3.bat`

Procedure

1. Start the MobiLink server.
   a. At a command prompt, navigate to *%SQLANYSAMP17%*`\MobiLink\Contact`.
   b. Run the following command:

      ```
      step1 -pwd=sql
      ```

      This command runs a batch file that starts the MobiLink server in a verbose mode. This mode is useful during development or troubleshooting, but has a significant performance impact and so would not be used in a routine production environment.
2. Synchronize both remote databases.
   a. At a command prompt, navigate to *%SQLANYSAMP17%*`\MobiLink\Contact`.
   b. Run the following command:

      ```
      step2
      ```

      This is a batch file that synchronizes both remote databases.
3. Shut down the MobiLink server.
   a. At a command prompt, navigate to *%SQLANYSAMP17%*`\MobiLink\Contact`.
   b. Run the following command:

      ```
      step3
      ```

This is a batch file that shuts down the MobiLink server.

Results

The Contact sample is run and the remote and consolidated databases are synchronized.

Next Steps

To explore how synchronization works in the Contact sample, you can use Interactive SQL to modify the data in the remote and consolidated databases, and use the batch files to synchronize.

1.4.2 Tables in the Contact Databases

The table definitions for the Contact database are located in the `MobiLink\Contact\build_consol.sql` and `MobiLink\Contact\build_remote.sql` files, under your samples directory.

Both the consolidated and the remote databases contain the following three tables, although their definition is slightly different in each place.

SalesRep

Each sales representative occupies one row in the SalesRep table. Each remote database belongs to a single sales representative.

In each remote database, SalesRep has the following columns:

rep_id

A primary key column that contains an identifying number for the sales representative.

name

The name of the representative.

In the consolidated database only, there is also an ml_username column holding the MobiLink user name for the representative.

Customer

This table holds one row for each customer. Each customer is a company with which a single sales representative does business. There is a one-to-many relationship between the SalesRep and Customer tables.

In each remote database, Customer has the following columns:

cust_id

A primary key column holding an identifying number for the customer.

name

The customer name. This is a company name.

rep_id

A foreign key column that references the SalesRep table. Identifies the sales representative assigned to the customer.

In the consolidated database, there are two additional columns, last_modified and active:

last_modified

The last time the row was modified. This column is used for timestamp-based synchronization.

active

A BIT column that indicates if the customer is currently active (1) or if the company no longer deals with this customer (0). If the column is marked inactive (0) all rows corresponding to this customer are deleted from remote databases.

Contact

This table holds one row for each contact. A contact is a person who works at a customer company. There is a one-to-many relationship between the Customer and Contact tables.

In each remote database, Contact has the following columns:

contact_id

A primary key column holding an identifying number for the contact.

name

The name of the individual contact.

cust_id

The identifier of the customer for whom the contact works.

In the consolidated database, the table also has the following columns:

last_modified

The last time the row was modified. This column is used for timestamp-based synchronization.

active

A BIT column that indicates if the contact is currently active (1) or if the company no longer deals with this contact (0). If the column is marked inactive (0) the row corresponding to this contact is deleted from remote databases.

Product

Each product sold by the company occupies one row in the Product table. The Product table is held in a separate publication so that remote databases can synchronize the table separately.

In each remote database, Product has the following columns:

id

A primary key column that contains a number to identify the product.

name

The name of the item.

size

The size of the item.

quantity

The number of items in stock. When a sales representative takes an order, this column is updated.

unit_price

The price per unit of the product.

In the consolidated database, the Product table has the following additional columns:

supplier

The company that manufactures the product.

last_modified

The last time the row was modified. This column is used for timestamp-based synchronization.

active

A BIT column that indicates if the product is currently active (1). If the column is marked inactive (0), the row corresponding to this product is deleted from remote databases.

In addition to these tables, a set of tables is created at the consolidated database only. These include the product_conflict table, which is a temporary table used during conflict resolution, and a set of tables for monitoring MobiLink activities owned by a user named mlmaint. Scripts to create the MobiLink monitoring tables are in the file *%SQLANYSAMP17%*`\MobiLink\Contact\mlmaint.sql`.

1.4.3 Users in the Contact Sample

The Contact sample includes several different database user IDs and MobiLink user names.

Database User IDs

The two remote databases are assigned to the sales representatives Samuel Singer (rep_id 856) and Pamela Savarino (rep_id 949).

When connecting to their remote database, both users use the default SQL Anywhere user ID **`dba`** and the password **`SQL`**.

Each remote database also has a user ID **`sync_user`** with the password **`sync_user`**. This user ID is employed only on the dbmlsync command line. The **`sync_user`** must have the SYS_RUN_REPLICATION_ROLE system role and can perform any operation when connected from dbmlsync, but has no authority when connected from any other application. So, using the **`sync_user`** ID and password should not be a problem.

At the consolidated database, there is a user named **`mlmaint`**, who owns the tables for monitoring MobiLink synchronization statistics and errors. The **`mlmaint`** user has no right to connect. The assignment of the tables to a separate user ID is done simply to separate the objects from the others in the schema for easier administration in SQL Central and other utilities.

MobiLink User Names

MobiLink user names are distinct from database user IDs. Each remote device has a MobiLink user name in addition to the user ID they use when connecting to a database. The MobiLink user name for Samuel Singer is SSinger. The MobiLink user name for Pamela Savarino is PSavarino. The MobiLink user name is stored or used in the following locations:

- At the remote database, the MobiLink user name is added using a CREATE SYNCHRONIZATION USER statement.
- At the consolidated database, the MobiLink user name and password are added using the mluser utility.
- During synchronization, the MobiLink password for the connecting user is supplied on the dbmlsync command line listed in `MobiLink\Contact\step2.bat`.
- The MobiLink server supplies the MobiLink user name as a parameter to many of the scripts during synchronization.
- The SalesRep table at the consolidated database has an ml_username column. The synchronization scripts match the MobiLink user name parameter against the value in this column.

1.4.4 Synchronization of Sales Representatives in the Contact Sample

The synchronization scripts for the SalesRep table illustrate **snapshot synchronization**. Regardless of whether a sales representative's information has changed, it is downloaded.

Business Rules

The business rules for the SalesRep table are as follows:

- The table must not be modified at the remote database.
- A sales representative's MobiLink user name and rep_id value must not change.

- Each remote database contains a single row from the SalesRep table, corresponding to the remote database owner's MobiLink user name.

Downloads

download_cursor

At each remote database, the SalesRep table contains a single row. There is very little overhead for the download of a single row, so a simple snapshot download_cursor script is used:

```
SELECT rep_id, name
FROM SalesRep
WHERE ? IS NOT NULL
AND ml_username = ?
```

The first parameter in the script is the last download timestamp, which is not used. The IS NOT NULL expression is a dummy expression supplied to use the parameter. The second parameter is the MobiLink user name.

Uploads

This table should not be updated at the remote database, so there are no upload scripts for the table.

Related Information

Snapshot Synchronization

1.4.5 Synchronization of Customers in the Contact Sample

The synchronization scripts for the Customer table illustrate **timestamp-based synchronization** and partitioning rows. Both of these techniques minimize the amount of data that is transferred during synchronization while maintaining consistent table data.

Business Rules

The business rules governing customers are as follows:

- Customer information can be modified at both the consolidated and remote databases.
- Periodically, customers may be reassigned among sales representatives. This process is commonly called territory realignment.

- Each remote database contains only the customers they are assigned to.

Downloads

download_cursor

The following download_cursor script downloads only active customers for whom information has changed since the last successful download. It also filters customers by sales representative.

```
SELECT cust_id, Customer.name, Customer.rep_id
FROM Customer key join SalesRep
WHERE Customer.last_modified >= ?
AND SalesRep.ml_username = ?
AND Customer.active = 1
```

download_delete_cursor

The following download_delete_cursor script downloads only customers for whom information has changed since the last successful download. It deletes all customers marked as inactive or who are not assigned to the sales representative.

```
SELECT cust_id
FROM Customer key join SalesRep
WHERE Customer.last_modified >= ?
AND ( SalesRep.ml_username != ? OR Customer.active = 0 )
```

If rows are deleted from the Customer table at the consolidated database, they do not appear in this result set and so are not deleted from remote databases. Instead, customers are marked as inactive.

When territories are realigned, this script deletes those customers no longer assigned to the sales representative. It also deletes customers who are transferred to other sales representatives. Such additional deletes are flagged with a SQLCODE of 100 but do not interfere with synchronization. A more complex script could be developed to identify only those customers transferred away from the current sales representative.

The MobiLink client performs cascading deletes at the remote database, so this script also deletes all contacts who work for customers assigned to some other sales representative.

Uploads

Customer information can be inserted, updated, or deleted at the remote database. The scripts corresponding to these operations are as follows:

upload_insert

The following upload_insert script adds a row to the Customer table, marking the customer as active:

```
INSERT INTO Customer(
 cust_id, name, rep_id, active )
VALUES ( ?, ?, ?, 1 )
```

upload_update

The following upload_update script modifies the customer information at the consolidated database. Conflict detection is not done on this table.

```
UPDATE Customer
SET name = ?, rep_id = ?
WHERE cust_id = ?
```

upload_delete

The following upload_delete script marks the customer as inactive at the consolidated database. It does not delete a row.

```
UPDATE Customer
SET active = 0
WHERE cust_id = ?
```

Related Information

Partitioned Rows Among Remote Databases
Implementing Timestamp-based Downloads

1.4.6 Synchronization of Contacts in the Contact Sample

The Contact table contains the name of a person working at a customer company, a foreign key to the customer, and a unique integer identifying the contact. It also contains a last_modified timestamp and a marker to indicate whether the contact is active.

Business Rules

The business rules for this table are as follows:

- Contact information can be modified at both the consolidated and remote databases.
- Each remote database contains only those contacts who work for customers they are assigned to.
- When customers are reassigned among sales representatives, contacts must also be reassigned.

Trigger

A trigger on the Customer table is used to ensure that the contacts get picked up when information about a customer is changed. The trigger explicitly alters the last_modified column of each contact whenever the corresponding customer is altered:

```
CREATE TRIGGER UpdateCustomerForContact
AFTER UPDATE OF rep_id ORDER 1
```

```
ON DBA.Customer
REFERENCING OLD AS old_cust NEW as new_cust
FOR EACH ROW
BEGIN
  UPDATE Contact
  SET Contact.last_modified = new_cust.last_modified
  FROM Contact
  WHERE Contact.cust_id = new_cust.cust_id
END
```

By updating all contact records whenever a customer is modified, the trigger ties the customer and their associated contacts together. Whenever a customer is modified, all associated contacts are modified too, and the customer and associated contacts are downloaded together on the next synchronization.

Downloads

download_cursor

The download_cursor script for Contact is as follows:

```
SELECT contact_id, contact.name, contact.cust_id
FROM ( contact JOIN customer ) JOIN salesrep
ON contact.cust_id = customer.cust_id
 AND customer.rep_id = salesrep.rep_id
WHERE Contact.last_modified >= ?
 AND salesrep.ml_username = ?
 AND Contact.active = 1
```

This script retrieves all contacts that are active, that have been changed since the last time the sales representative downloaded (either explicitly or by modification of the corresponding customer), and that are assigned to the representative. A join with the Customer and SalesRep table is needed to identify the contacts associated with this representative.

download_delete_cursor

The download_delete_cursor script for Contact is as follows:

```
SELECT contact_id
FROM ( Contact JOIN Customer ) JOIN SalesRep
ON Contact.cust_id = Customer.cust_id
 AND Customer.rep_id = SalesRep.rep_id
WHERE Contact.last_modified >= ?
 AND Contact.active = 0
```

The automatic use of cascading referential integrity by the MobiLink client deletes contacts when the corresponding customer is deleted from the remote database. The download_delete_cursor script therefore has to delete only those contacts marked as inactive.

Uploads

Contact information can be inserted, updated, or deleted at the remote database. The scripts corresponding to these operations are as follows:

upload_insert

The following upload_insert script adds a row to the Contact table, marking the contact as active:

```
INSERT INTO Contact (
 contact_id, name, cust_id, active )
VALUES ( ?, ?, ?, 1 )
```

upload_update

The following upload_update script modifies the contact information at the consolidated database:

```
UPDATE Contact
SET name = ?, cust_id = ?
WHERE contact_id = ?
```

Conflict detection is not done on this table.

upload_delete

The following upload_delete script marks the contact as inactive at the consolidated database. It does not delete a row.

```
UPDATE Contact
SET active = 0
WHERE contact_id = ?
```

1.4.7 Synchronization of Products in the Contact Sample

The scripts for the Product table illustrate conflict detection and resolution.

The Product table is kept in a separate publication from the other tables so that it can be downloaded separately. For example, if the price changes and the sales representative is synchronizing over a slow link, they can download the product changes without uploading their own customer and contact changes.

Business Rules

The only change that can be made at the remote database is to change the quantity column, when an order is taken.

Downloads

download_cursor

The following download_cursor script downloads all rows changed since the last time the remote database synchronized:

```
SELECT id, name, size, quantity, unit_price
FROM product
WHERE last_modified >= ?
AND active = 1
```

download_delete_cursor

The following download_delete_cursor script removes all products no longer sold by the company. These products are marked as inactive in the consolidated database.

```
SELECT id, name, size, quantity, unit_price
FROM product
WHERE last_modified >= ?
AND active = 0
```

Uploads

Only UPDATE operations are uploaded from the remote database. The major feature of these upload scripts is a conflict detection and resolution procedure.

If two sales representatives take orders and then synchronize, each order is subtracted from the quantity column of the Product table. For example, if Samuel Singer takes an order for 20 baseball hats (product ID 400), he changes the quantity from 90 to 70. If Pamela Savarino takes an order for 10 baseball hats before receiving this change, she changes the column in her database from 90 to 80.

When Samuel Singer synchronizes his changes, the quantity column in the consolidated database is changed from 90 to 70. When Pamela Savarino synchronizes her changes, the correct action is to set the value to 60. This setting is accomplished by detecting the conflict.

The conflict detection scheme includes the following scripts:

upload_update

The following upload_update script is a straightforward UPDATE at the consolidated database:

```
UPDATE product
SET name = ?, size = ?, quantity = ?, unit_price = ?
WHERE product.id = ?
```

upload_fetch

The following upload_fetch script fetches a single row from the Product table for comparison with the old values of the uploaded row. If the two rows differ, a conflict is detected.

```
SELECT id, name, size, quantity, unit_price
FROM Product
WHERE id = ?
```

upload_old_row_insert

If a conflict is detected, the old values are placed into the product_conflict table for use by the resolve_conflict script. The row is added with a value of O (for Old) in the row_type column.

```
INSERT INTO DBA.product_conflict(
 id, name, size, quantity, unit_price, row_type )
VALUES( ?, ?, ?, ?, ?, 'O' )' )
```

upload_new_row_insert

The following script adds the new values of the uploaded row into the product_conflict table for use by the resolve_conflict script:

```
INSERT INTO DBA.product_conflict(
 id, name, size, quantity, unit_price, row_type )
VALUES( ?, ?, ?, ?, ?, 'N' )
```

Conflict Resolution

resolve_conflict

The following script resolves the conflict by adding the difference between new and old rows to the quantity value in the consolidated database:

```
UPDATE Product
SET p.quantity = p.quantity
              - old_row.quantity
              + new_row.quantity
FROM Product p,
     DBA.product_conflict old_row,
     DBA.product_conflict new_row
WHERE p.id = old_row.id
    AND p.id = new_row.id
    AND old_row.row_type = 'O'
    AND new_row.row_type = 'N'
```

1.4.8 Statistic and Error Monitoring in the Contact Sample

The Contact sample contains some simple error reporting and monitoring scripts. The SQL statements to create these scripts are in the file `MobiLink\Contact\mlmaint.sql`.

The scripts insert rows into tables created to hold the values. For convenience, the tables are owned by a distinct user, mlmaint.

1.5 MobiLink Tutorials

The following tutorials range from introductory tutorials for news users to demonstrations of how to use advanced features.

Additional MobiLink tutorials are available online.

> **i Note**
>
> The online tutorials are based on version 12.0.0 of SQL Anywhere. Some visuals and procedures may differ from SQL Anywhere 17.0.11.

In this section:

Tutorial: Introducing MobiLink [page 106]
    This tutorial guides you through the basic steps for writing synchronization scripts, interpreting MobiLink logs, and monitoring synchronizations between a consolidated database and two remote

databases using the MobiLink Profiler. It provides instructions for setting up the databases and synchronizations using SQL Central.

Tutorial: Using MobiLink with a SQL Anywhere Consolidated Database [page 129]
: This tutorial shows you how to mobilize a SQL Anywhere database by using MobiLink. It sets up synchronization between a SQL Anywhere consolidated database and an UltraLite remote database. You can also use a SQL Anywhere remote database.

Tutorial: Using MobiLink with an Oracle Database 11g [page 143]
: This tutorial shows you to mobilize an Oracle Database 10g using MobiLink. It sets up synchronization between an Oracle Database 11g and a SQL Anywhere remote database. You could also set up an UltraLite remote database.

Tutorial: Using MobiLink with an Adaptive Server Enterprise Consolidated Database [page 165]
: This tutorial shows you how to mobilize an Adaptive Server Enterprise database using MobiLink. It sets up synchronization between an Adaptive Server Enterprise consolidated database and a SQL Anywhere remote database. You could also use UltraLite clients.

Tutorial: Using Java or .NET for Custom User Authentication [page 188]
: MobiLink synchronization scripts can be written in SQL, Java, or .NET. You can use Java or .NET to add custom actions at any point of a synchronization.

Tutorial: Using Direct Row Handling [page 198]
: You can use direct row handling to communicate remote data to any central data source, application, or web service other than a supported consolidated database.

Tutorial: Synchronizing with Microsoft Excel [page 225]
: You can use direct row handling to communicate remote data to any central data source, application, or web service.

Tutorial: Synchronizing with XML [page 247]
: This tutorial shows you how to synchronize data in an XML file to remote clients.

Tutorial: Using Central Administration of Remote Databases [page 269]
: This tutorial leads you through the process of setting up central administration of remote databases and demonstrates how several common operations can be performed.

Tutorial: Changing a Schema Using the Script Version Clause [page 300]
: This tutorial describes how to perform a schema change on a remote database involved in synchronization where the dbmlsync ScriptVersion extended option is not being used.

Tutorial: Changing a Schema Using the ScriptVersion Extended Option [page 311]
: This tutorial demonstrates how to perform a schema change when you are using the ScriptVersion extended option.

Tutorial: Simulating Multiple MobiLink Clients Using the MobiLink Replay Utility [page 319]
: This tutorial demonstrates how to use the mlreplay utility to simulate multiple MobiLink clients on a single computer.

1.5.1 Tutorial: Introducing MobiLink

This tutorial guides you through the basic steps for writing synchronization scripts, interpreting MobiLink logs, and monitoring synchronizations between a consolidated database and two remote databases using the MobiLink Profiler. It provides instructions for setting up the databases and synchronizations using SQL Central.

Prerequisites

You require a basic knowledge of MobiLink event scripts

The following software is required:

- SQL Anywhere 17

You must have the following roles and privileges on the consolidated database:

- SYS_AUTH_DBA_ROLE compatibility role

You must have the following roles and privileges on the remote database:

- SYS_AUTH_DBA_ROLE compatibility role
- SYS_RUN_REPLICATION_ROLE system role

Context

This tutorial shows you how to:

- Migrate a consolidated database schema to remote databases
- Create the basic scripts needed for synchronization and store them in the consolidated database using SQL Central
- Start the MobiLink server
- Monitor synchronization using log files and the MobiLink Profiler

1. Lesson 1: Setting up a MobiLink Consolidated Database [page 108]
   In this lesson, you set up a SQL Anywhere consolidated database by creating it and defining an ODBC data source.
2. Lesson 2: Creating and Populating a Table in the MobiLink Consolidated Database [page 109]
   In this lesson, you create the `Product` table and insert sample data in the MobiLink consolidated database.
3. Lesson 3: Creating a MobiLink Project and Synchronization Model [page 111]
   In this lesson, you use the *Create Project Wizard* to create a new MobiLink project. The *Create Project Wizard* also creates a synchronization model using the defaults, which can be edited later.
4. Lesson 4: Testing Your Synchronization Model [page 113]
   This lesson demonstrates how to quickly test your synchronization model.
5. Lesson 5: Refining a Synchronization Model [page 114]

This lesson shows you how to make changes to your synchronization model and explores some of the choices available to you when developing a model.
6. Lesson 6: Choosing MobiLink Server Options [page 116]
   This lesson shows you how to choose options that are later used to run the MobiLink server when you deploy your synchronization model.
7. Lesson 7: Deploying the Synchronization Model [page 118]
   In this lesson, you deploy the synchronization model using the *Deploy Synchronization Model Wizard* to configure the consolidated database for synchronization, and create and deploy the remote database.
8. Lesson 8: Starting the MobiLink Server [page 119]
   In this lesson, you start the MobiLink server using the mlsrv17 -c option to connect to your consolidated database. You use additional options to configure MobiLink server behavior.
9. Lesson 9: Starting the MobiLink Clients [page 120]
   In this lesson, you start the remote database to prepare it for synchronization. This lesson assumes that your remote database, consolidated database, and MobiLink server reside on the same computer.
10. Lesson 10: Starting the MobiLink Profiler [page 121]
   In this lesson, you start and configure the MobiLink Profiler to see synchronizations as they occur.
11. Lesson 11: Synchronizing [page 122]
   In this lesson, you synchronize the remote database with the consolidated database using the dbmlsync utility.
12. Lesson 12: Using the MobiLink Server Log File Viewer to Check for Errors and Warnings [page 124]
   After the tables are synchronized, you can view the progress of the synchronizations using the message log files you created with each command line, `mlsrv.mls` and `remote.dbs` respectively. The default location of these files is the directory where the command was run.
13. Lesson 13: Monitoring MobiLink Resources with the SQL Anywhere Monitor [page 125]
   In this lesson, you set up monitoring of a MobiLink server and a MobiLink server farm. This lesson uses the Monitor Developer Edition.
14. Lesson 14: Cleaning Up [page 128]
   Remove all tutorial materials from your computer.

Related Information

Interactive SQL
SQL Central
MobiLink Synchronization [page 4]
Synchronization Scripts
Synchronization Techniques
Synchronization Events
MobiLink Profiler
Conflict Handling Overview
SAP SQL Anywhere Forum

1.5.1.1 Lesson 1: Setting up a MobiLink Consolidated Database

In this lesson, you set up a SQL Anywhere consolidated database by creating it and defining an ODBC data source.

Procedure

1. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Central*.
2. Click *Tools* > *SQL Anywhere 17* > *Create Database*.
3. Click *Next*.
4. Leave the default of *Create a database on this computer*, and click *Next*.
5. In the *Save the main database file to the following file* field, type `c:\MLintro\MLconsolidated.db`. Click *Next*. If the directory does not exist, you are asked to create it. Click *Yes*.
6. Follow the remaining instructions in the *Create Database Wizard* and accept the default values. If prompted to specify a user ID and password for the DBA user, enter **`DBA`** and **`passwd`**, respectively.

   On the *Connect To The Database* page, check the *Stop database after last disconnect* option.
7. Click *Finish*.

   The **`MLconsolidated`** database is created.
8. In the *Creating Database* window, click *Close* if the window did not close automatically.
9. Click *Connections* > *Disconnect* to stop the personal database server.
10. Click *Tools* > *SQL Anywhere 17* > *Open ODBC Administrator*.
11. Click the *User DSN* tab, and click *Add*.
12. In the *Create New Data Source* window, click *SQL Anywhere 17*, and click *Finish*.
13. Perform the following tasks in the *ODBC Configuration For SQL Anywhere* window:
    a. Click the *ODBC* tab.
    b. In the *Data source name* field, type **`mlintro_consdb`**.
    c. Click the *Login* tab.
    d. In the *Authentication* dropdown list, leave the default of *Database* to connect using your user ID and password.
    e. In the *User ID* field, type **`DBA`**.
    f. In the *Password* field, type **`passwd`**.
    g. In the *Action* dropdown list, leave the default of *Start and connect to a database on this computer*.
    h. In the *Database file* field, type `c:\MLintro\MLconsolidated.db`.
    i. In the *Server name* field, type **`MLconsolidated`**.
    j. In the *Start line* field type dbsrv17.
    k. Click the *ODBC* tab and click the *Test Connection* button to verify that the connection is successful. Click *OK*.
    l. Click *OK* and then click *OK* again to close the *ODBC Data Source Administrator* window.

14. Click *Connections* ▸ *Connect with SQL Anywhere 17*.
15. In the *Action* dropdown list, select *Connect with an ODBC data source*.
16. In the *ODBC Data Source Name* field, type `mlintro_consdb`.
17. Click *Connect*.

Results

A SQL Anywhere consolidated database and an ODBC data source are created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Introducing MobiLink [page 106]

Next task: Lesson 2: Creating and Populating a Table in the MobiLink Consolidated Database [page 109]

Related Information

MobiLink Consolidated Databases
Initialization Utility (dbinit)

1.5.1.2 Lesson 2: Creating and Populating a Table in the MobiLink Consolidated Database

In this lesson, you create the `Product` table and insert sample data in the MobiLink consolidated 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. Connect to the consolidated database in Interactive SQL. At a command prompt, run the following command:

   You can start Interactive SQL from SQL Central or at a command prompt.

   - From SQL Central, right-click the *MLconsolidated - DBA* database and click *Open Interactive SQL*.

   -
     ```
     dbisql -c "DSN=mlintro_consdb"
     ```
2. Execute the following SQL statement in Interactive SQL to create the **`Product`** table:

   ```
   CREATE TABLE Product (
   name VARCHAR(128) NOT NULL PRIMARY KEY,
   quantity INTEGER
   );
   ```

   The **`Product`** table contains the following columns:

Column	Description
name	The name of the product.
quantity	The number of items sold.

   After creating the tables, you populate the **`Product`** table with sample data.
3. Execute the following SQL statements in Interactive SQL to populate the **`Product`** table with sample data:

   ```
   INSERT INTO Product(name, quantity)
       VALUES ( 'Screwmaster Drill', 10);
   INSERT INTO Product(name, quantity)
       VALUES ( 'Drywall Screws 10lb', 30);
   INSERT INTO Product(name, quantity)
       VALUES ( 'Putty Knife x25', 12);
   COMMIT;
   ```
4. Verify that the **`Product`** table contains the data inserted from the previous step.

   Execute the following SQL statement to verify the contents:

   ```
   SELECT * FROM Product
   ```

   The contents of the **`Product`** table should appear in Interactive SQL.
5. Close Interactive SQL. You do not need to save your SQL statements.

Results

The Products table is created in the consolidated database.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Introducing MobiLink [page 106]

Previous task: Lesson 1: Setting up a MobiLink Consolidated Database [page 108]

Next task: Lesson 3: Creating a MobiLink Project and Synchronization Model [page 111]

Related Information

Interactive SQL
CREATE TABLE Statement

1.5.1.3 Lesson 3: Creating a MobiLink Project and Synchronization Model

In this lesson, you use the *Create Project Wizard* to create a new MobiLink project. The *Create Project Wizard* also creates a synchronization model using the defaults, which can be edited later.

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. From SQL Central, click *Tools* > *MobiLink 17* > *New Project*.

   The *Create Project Wizard* appears.
2. In the *Name* field, type **`mlintro_project`**.
3. In the *Location* field, type `C:\MLintro`. Click *Next*.
4. In the *Database display name* field, type **`mlintro_consdb`**.
5. In the *Connection string* field type **`dsn=mlintro_consdb`** and then click *Next*.
6. Ensure that the *Product* table is selected from the *Which consolidated database tables and columns do you want to have in your remote database* list, and then click *Next*.

7. Select the *Add a remote schema name to the project* option.
8. In the *What do you want to name the new remote schema* field, type **`sync_mlintro`** and click *Next*.
9. Select *SQL Anywhere* for the *Which type of remote database do you want to use* option and click *Finish*.
10. Click *Yes* to install the MobiLink system tables, and then click *OK*.

    A synchronization model with the same name as your remote schema is created.
11. Select the *Product* table row from the *Mappings* tab.
12. On the *Download Strategy* tab in the *Details* pane, select *Timestamp* for the *Strategy*.
13. Select *Store timestamp column in a shadow table*.

    Using shadow tables is often preferred because it does not require any changes to existing tables. In contrast, the default setting adds a timestamp column to your table and typically has better performance.
14. To save the changes to the synchronization model, click *File* > *Save*.

Results

The MobiLink project, a remote schema name, and a synchronization model are created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Introducing MobiLink [page 106]

Previous task: Lesson 2: Creating and Populating a Table in the MobiLink Consolidated Database [page 109]

Next task: Lesson 4: Testing Your Synchronization Model [page 113]

Related Information

Synchronization Models [page 37]
Consolidated Database Setup
MobiLink Server System Tables
MobiLink Server System Procedures
download_delete_cursor Scripts
Conflict Handling Overview
Conflict Resolution
Publications
Synchronization Model Tasks [page 41]
Table and Column Mappings [page 42]

Modifying the Download Strategy [page 47]
Modifying Conflict Detection and Resolution [page 55]

1.5.1.4 Lesson 4: Testing Your Synchronization Model

This lesson demonstrates how to quickly test your synchronization model.

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 under *MobiLink 17*, expand **`mlintro_project`**. Click *Synchronization Models* and then choose **`sync_mlintro`**.
2. Click *File* > *Test*.
3. Click *OK* to dismiss the warning indicating that your consolidated database will be modified.

   The synchronization model is deployed to the consolidated database and a remote database is created for testing purposes. The MobiLink server is started.
4. Click the *Data* tab. The top pane shows rows from the Product table in the consolidated database. The bottom pane shows that the remote database currently contains no rows in the Product table.

   If more than one table were defined, the *Show* dropdown list would let you select which table to view.
5. Click *Synchronize*. The three rows now appear in the remote database.
6. Select the *Client Log* tab. Scan the log for error messages or warnings. Do the same on the *MobiLink Log* tab.
7. Click the *Data* tab. In the lower pane, right-click the row with the name **`Drywall Screws 10lb`**, select *Edit Row*, and then change the quantity to 99.
8. Click *Synchronize*. The change made on the remote database has been synchronized to the consolidated database.
9. Click the *Actions* button and then select *Open Interactive SQL on the consolidated database*. The Interactive SQL window opens with a connection to the consolidated database.
10. Execute the following SQL statements and then close Interactive SQL:

    ```
    update product set quantity = quantity + 1;
    commit;
    ```

11. Back in the *Data* tab in SQL Central, click *Actions* > *Refresh Data Tab*. You can see the updated rows in the consolidated database.

12. Click *Synchronize*. The remote database is updated with the new quantity values.
13. Close the *Test* window.

Results

You have successfully tested the synchronization model.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Introducing MobiLink [page 106]

Previous task: Lesson 3: Creating a MobiLink Project and Synchronization Model [page 111]

Next task: Lesson 5: Refining a Synchronization Model [page 114]

Related Information

Synchronization Model Deployment [page 58]

1.5.1.5 Lesson 5: Refining a Synchronization Model

This lesson shows you how to make changes to your synchronization model and explores some of the choices available to you when developing a model.

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 under *MobiLink 17*, expand **`mlintro_project`**. Click *Synchronization Models* and then choose **`sync_mlintro`**.
2. In the *Mappings* tab, select the row for the Product table.
3. In the lower pane, select the *Conflict Handling* tab.
4. For *Choose a strategy for resolving conflicts when they are detected*, select *First in wins*. Selecting this option means that if a row is modified on both the consolidated and remote databases, the value in the consolidated database is considered to be the correct one.
5. In the *Deployment* tab, click *Test*. If you are prompted to save your synchronization model, click *Yes*.
6. Select the *Data* tab and then click *Synchronize* to update the remote database.
7. Edit the row named *Screwmaster Drill* and set the quantity on the consolidated database to 20 and the quantity on the remote database to 10.
8. Click *Synchronize*. Both the remote and consolidated databases should show a quantity of 20.
9. Close the *Test* window.
10. Select the *Events* tab for the **`sync_mlintro`** model. This page shows the SQL statements that the MobiLink server runs for this synchronization model. The green bars in the left margin indicate that the SQL statements are automatically generated based on the choices you made in the *Mappings* tab.

    There may be times when the options available in the *Mappings* tab are not sufficient for your scenario. In these cases, you can further customize your synchronization scripts using the *Events* tab.
11. In the *Events* tab, look for the upload_insert event for the Product table. This shows the SQL statements that the MobiLink server executes when it receives a new row from a remote database. We'll make a change to this event to limit any new orders to a maximum quantity of 50. The text **{ml r."quantity"}** represents the remote quantity column that was uploaded. Change this to the following:

    ```
    If {ml r."quantity"} < 50 then {ml r."quantity"} else 50 end if
    ```

    The full upload_insert event should now be as follows:

    ```
    Product (DBA): upload_insert
    /* Insert the row into the consolidated database. */
    INSERT INTO "DBA"."Product" ( "name", "quantity" )
    VALUES ( {ml r."name"}, If {ml r."quantity"} < 50 then {ml r."quantity"} else
    50 end if )
    ```

    The bar in the margin is now yellow for the upload_insert event, indicating that the settings from the *Mappings* tab have been overridden.
12. From the *Deployment* tab, click *Test*. Select the *Data* tab, and then click *Synchronize*. Now right-click in the lower pane for the remote database. Select *Add Row* then add a row with a name of **`Hammer`** and a quantity of **`200`**. Click *Synchronize*. Now both the consolidated and remote database contain a value of 50 for the **`Hammer`** row.
13. Close the *Test* window and return the *Events* tab. Right-click the upload_insert event script and then select *Restore 'Product (DBA): upload_insert' Script*. The bar in the margin is now green again, and the customizations made previously have been undone.

Results

Changes are made to the synchronization model.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Introducing MobiLink [page 106]

Previous task: Lesson 4: Testing Your Synchronization Model [page 113]

Next task: Lesson 6: Choosing MobiLink Server Options [page 116]

Related Information

Synchronization Model Deployment [page 58]

1.5.1.6 Lesson 6: Choosing MobiLink Server Options

This lesson shows you how to choose options that are later used to run the MobiLink server when you deploy your synchronization model.

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 under *MobiLink 17*, expand **`mlintro_project`** and then click *MobiLink Server Command Lines*. If you are prompted to save your synchronization model, click *Yes*.
2. In the right pane, right click *Default* and select *Properties*.
3. In the *Default MobiLink Server Command Line Properties* window, perform the following tasks:

- Select the *General* tab. From the *Verbosity* dropdown list, select *High (-v+)*.
- Select the *Advanced* tab
- Set the -dl option to *true*.
- Set the -o option to *mlsrv.mls*.
- Set the -zf option to *true*.
- Set the -zu option to *true*.

> i Note
>
> The -zf option should be used for debugging and development purposes only. This tutorial requires the -zf option so that you do not need to shut down the MobiLink server when adding new scripts to the consolidated database in a later lesson. The -zu+ option automatically adds new MobiLink users to the synchronization environment.

4. Click *OK*.

Results

The options for running the MobiLink server when the synchronization model is deployed have been set.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Introducing MobiLink [page 106]

Previous task: Lesson 5: Refining a Synchronization Model [page 114]

Next task: Lesson 7: Deploying the Synchronization Model [page 118]

Related Information

Synchronization Model Deployment [page 58]

1.5.1.7 Lesson 7: Deploying the Synchronization Model

In this lesson, you deploy the synchronization model using the *Deploy Synchronization Model Wizard* to configure the consolidated database for synchronization, and create and deploy the remote 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. In the left pane of SQL Central under *MobiLink 17*, expand **`mlintro_project`**. Click *Synchronization Models* and then choose **`sync_mlintro`**.
2. Click *File* > *Deploy*.
3. Accept the default setting for the *Select the folder to contain files generated by the Wizard* field and click *Next*.
4. Accept the default settings for the *Client Network Options* page and click *Next*.
5. On the *MobiLink User and Password* page under *What MobiLink User And Password Do You Want To Use*, select *Use These* and perform the following tasks:
   a. In the *MobiLink user* field, type **`mlintro_user`**.
   b. In the *MobiLink password* field, type **`passwd`**.
   c. Check *Register this user in the consolidated database. Registered users are permitted to synchronize* and click *Next*.
6. On the *Synchronization Profile* page, type **`mlintro_remote_syncprofile`** in the *Synchronization profile name* field. Click *Browse* and set *Verbosity* to *high*. Click *OK* and then click *Next*.
7. Perform the following tasks on the *Choose How To Prepare Databases For Synchronization* page:
   a. For *What do you want to do with the SQL script created to prepare the consolidated database for synchronization*, select *Execute against consolidated database*.
   b. For *What do you want to do with the SQL script created to prepare the remote database for synchronization*, select *Execute against a new remote database*
   c. In the *User ID* field, type **`DBA`**.
   d. In the *Password* and field *Retype Password* fields, type **`passwd`**.
   e. Click *Next*.
8. On the *Review Your Choices* page, you can review the choice you made in the wizard. You can also review the SQL statements that will be executed against your databases by using the *View* buttons. Click *Finish*.

   A remote database called `sync_mlintro_remote.db` is created.

Results

You have successfully created and deployed the remote database.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Introducing MobiLink [page 106]

Previous task: Lesson 6: Choosing MobiLink Server Options [page 116]

Next task: Lesson 8: Starting the MobiLink Server [page 119]

Related Information

Synchronization Model Deployment [page 58]
Using a SQL Anywhere Database as a Remote Database

1.5.1.8 Lesson 8: Starting the MobiLink Server

In this lesson, you start the MobiLink server using the mlsrv17 -c option to connect to your consolidated database. You use additional options to configure MobiLink server behavior.

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. At a command prompt, change to the `c:\MLintro\mlintro_project\sync_mlintro_deploy` directory.

2. Run the following command:

```
mlsrv.bat
```

> **i Note**
>
> The -zu option reduces the security of your server by allowing any user to connect and should not be used in a production environment.

Results

The MobiLink server starts.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Introducing MobiLink [page 106]

Previous task: Lesson 7: Deploying the Synchronization Model [page 118]

Next task: Lesson 9: Starting the MobiLink Clients [page 120]

1.5.1.9 Lesson 9: Starting the MobiLink Clients

In this lesson, you start the remote database to prepare it for synchronization. This lesson assumes that your remote database, consolidated database, and MobiLink server reside on the same computer.

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. At a command prompt, change to the `c:\MLintro\mlintro_project\sync_mlintro_deploy` directory.

2. Run the following command to start the **sync_mlintro_remote** database:

```
dbsrv17 sync_mlintro_remote
```

Results

The remote database is started.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Introducing MobiLink [page 106]

Previous task: Lesson 8: Starting the MobiLink Server [page 119]

Next task: Lesson 10: Starting the MobiLink Profiler [page 121]

1.5.1.10 Lesson 10: Starting the MobiLink Profiler

In this lesson, you start and configure the MobiLink Profiler to see synchronizations as they occur.

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

The MobiLink Profiler can be used to collect statistical information about synchronizations. The graphical chart shows tasks on the vertical axis against the progression of time on the horizontal axis. You can quickly identify synchronizations that result in errors or satisfy certain conditions using the MobiLink Profiler. Since the MobiLink Profiler does not significantly degrade performance, use it for both development and production environments.

Procedure

1. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *MobiLink Profiler*.
2. Connect the MobiLink Profiler to the MobiLink server. Click *File* > *Begin Profiling Session*.

   The *Connect to MobiLink Server* window appears.
3. In the *User* field, type `monitor_user`. A password is not required for this user.
4. In the *Host* field, type `localhost`. Click *OK*.

Results

This user is added automatically because you started the MobiLink server with the -zu+ option in a previous lesson. The Profiler begins collecting synchronization data.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Introducing MobiLink [page 106]

Previous task: Lesson 9: Starting the MobiLink Clients [page 120]

Next task: Lesson 11: Synchronizing [page 122]

Related Information

MobiLink Profiler

1.5.1.11 Lesson 11: Synchronizing

In this lesson, you synchronize the remote database with the consolidated database using the dbmlsync 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.

Procedure

1. At a command prompt, change to the `c:\MLintro\mlintro_project\sync_mlintro_deploy` directory.
2. Run the following command to synchronize the **`sync_mlintro_remote`** database:

   ```
   sync.bat "server=sync_mlintro_remote;UID=DBA;PWD=passwd"
   ```

   A window appears displaying all information relevant to the **`sync_mlintro_remote`** client synchronization with the consolidated database. The information is saved to the `remote.dbs` file, which is accessible after you close the client synchronization window.
3. Close the client synchronization window. Click *Shut Down*.
4. In the MobiLink Profiler, click the *Pause* button to stop the horizontal scrolling and then use the bottom pane to scroll back to the last synchronization.
5. Look at the properties of the last synchronization. Double-click the vertical colored bar to view the synchronization properties.

Results

The remote database is synchronized with the consolidated database.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Introducing MobiLink [page 106]

Previous task: Lesson 10: Starting the MobiLink Profiler [page 121]

Next task: Lesson 12: Using the MobiLink Server Log File Viewer to Check for Errors and Warnings [page 124]

Related Information

MobiLink SQL Anywhere Client Utility (dbmlsync) Syntax

1.5.1.12 Lesson 12: Using the MobiLink Server Log File Viewer to Check for Errors and Warnings

After the tables are synchronized, you can view the progress of the synchronizations using the message log files you created with each command line, `mlsrv.mls` and `remote.dbs` respectively. The default location of these files is the directory where the command was run.

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 SQL Central, click *Tools* › *MobiLink 17* › *MobiLink Server Log File Viewer*.
2. Open your message log file in a text editor and browse to `c:\MLintro\mlintro_project\sync_mlintro_deploy\mlsrv.mls`, and click *Open*.
   The *MobiLink Server Log File Viewer* window appears.
3. Click the *Synchronizations* tab to look for errors and warnings that occurred during synchronizations.
4. Click the *Messages* tab to look for errors and warnings that were reported by the MobiLink server.

   Clear the *Show Information* option, and click *Apply*.

   Only synchronizations that contain errors and warnings appear in the *Messages* pane. For example, you may see a warning that states the following:

   ```
   [10093] The MobiLink server is currently running with -zf that will reduce
   its performance
   ```
5. Click the *Summaries* tab to look for overall statistics listed in the message log file.
6. Open a client log file, such as `remote.dbs`, in a text editor.
7. Scan down the left side of the file. An error has occurred if you see a line that begins with an **E.**. Your synchronization has completed successfully if your log file does not contain errors.

Results

You have viewed the progress of the synchronizations using the message log files.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Introducing MobiLink [page 106]

Previous task: Lesson 11: Synchronizing [page 122]

Next task: Lesson 13: Monitoring MobiLink Resources with the SQL Anywhere Monitor [page 125]

Related Information

MobiLink Server Logging

1.5.1.13 Lesson 13: Monitoring MobiLink Resources with the SQL Anywhere Monitor

In this lesson, you set up monitoring of a MobiLink server and a MobiLink server farm. This lesson uses the Monitor Developer Edition.

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 Monitor. The following steps assume that the Monitor is not 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.

    In the *User Name* field, type **`admin`**, and in the *Password* field, type **`admin`**.

    > i Note
    >
    > You must be logged in to the Monitor as an administrator to perform the following steps. Read-only and operator users do not have the appropriate privileges to perform all the tasks.
    > 1. Log in to the Monitor.
    > 2. Click *Tools* > *User Settings* and review the *User Type* setting.

3. Add a MobiLink server resource to the Monitor:

    a. In the left navigation menu click *Tools* > *Administration*.
    b. Click *Resources*, and click *Add*.
    c. Click *MobiLink Server*, and click *Next*.
    d. In the *Name* field, type **`MobiLinkServerSample`**, and click *Next*.
    e. In the *Host* field, type **`localhost`**, and click *Next*.
    f. When you are prompted for the required authorization, in the *User ID* field, type a user name such as **`monitor_user`**, and in the *Password* field, type a password, such as **`passwd`**.

    These credentials are used to create a user on the MobiLink server. The Monitor stores this user ID and password and uses it to connect to the MobiLink server and monitor it.
    g. Click *Create*.
    h. The new resource, **`MobiLinkServerSample`**, is created and monitoring starts.
    i. Click *Close*.
    j. Click *Close*.
    k. Click *Overview* > *Resource List*. Click **`MobiLinkServerSample`** to create and open a dashboard for the resource.

4. Add a MobiLink server farm resource to monitor two MobiLink servers:

a. Add two MobiLink servers as resources to be monitored. For the first resource, use the *MobiLinkServerSample* resource that you added in the previous step.

   Add a second MobiLink server resource:

   1. At a command prompt, run the following command to start a MobiLink server that listens on port 8039:

      ```
      mlsrv17 -vcrs -zu+ -c "DSN=mlintro_consdb" -ot ml_tcpip.txt -zs
      ml_tcpip -x tcpip{port=8039}
      ```

   2. Click *Tools* > *Administration* .
   3. Click *Resources*, and click *Add*.
   4. Click *MobiLink Server*, and click *Next*.
   5. In the *Name* field, type **`ml_tcpip`**, and click *Next*.
   6. In the *Host* field, type **`localhost`**.
      In the *Port* field, type **`8039`**, and click *Next*.
   7. When you are prompted for the required authorization, in the *User ID* field, type a user name such as **`monitor_user`**, and in the *Password* field, type a password, such as **`passwd`**.
      These credentials are used to create a user on the MobiLink server. The Monitor stores this user ID and password and uses it to connect to the MobiLink server and monitor it.
   8. Click *Create*.
      The **`ml_tcpip`** resource is added to the *Resource List* in the *Overview* dashboard.
   9. Click *Close*.
   10. Click *Close*.

b. Add the MobiLink server farm resource.

   1. Open the *Administration* window.

      Click *Tools* > *Administration* .
   2. Click *Resources*, and click *Add*.
   3. Click *MobiLink Server Farm*, and click *Next*.
   4. In the *Name* field, type **`MobiLink_Test_Farm`**, and click *Next*.
   5. Click **`MobiLinkServerSample`** and **`ml_tcpip`**, and click *Create*.
   6. Click *Close*.
   7. Click *Close*.

c. Click *Dashboards* > *Overview* .

   The **`MobiLink_Test_Farm`** resource appears in the *Resource List*.

   The MobiLink server resources remain in the *Resource List*.

d. Click the arrow to the left of the **`MobiLink_Test_Farm`** to see the list of MobiLink server resources that are included in the farm.

e. Click **`MobiLink_Test_Farm`** to open the **`MobiLink_Test_Farm`** dashboard and view the collected metrics.

   The *Alert List*, *Resource Widget*, and *Server Info* widgets should appear.

5. To test an alert or learn about additional Monitor features, see Tutorial: Monitoring resources with the Monitor.

Results

You used the SQL Anywhere Monitor to monitor a MobiLink server and a MobiLink server farm.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Introducing MobiLink [page 106]

Previous task: Lesson 12: Using the MobiLink Server Log File Viewer to Check for Errors and Warnings [page 124]

Next task: Lesson 14: Cleaning Up [page 128]

Related Information

MobiLink Server in a Server Farm
Monitor Users

1.5.1.14 Lesson 14: Cleaning Up

Remove all tutorial materials from your computer.

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 completed the previous lessons in this tutorial.

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

Procedure

1. Close all instances of the following applications:
    - The MobiLink Profiler
    - The SQL Anywhere Monitor for MobiLink
    - SQL Central
    - Interactive SQL
2. Close any SQL Anywhere, MobiLink, and synchronization client windows.
3. Delete all tutorial-related data sources:
    a. Start the ODBC Data Source Administrator.
    b. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *ODBC Data Source Administrator*.
    c. Select **`mlintro_consdb`** from the list of *User Data Sources*, and click *Remove*.
4. Delete to the directory containing your consolidated and remote databases.

Results

The tutorial materials are removed from your computer.

Task overview: Tutorial: Introducing MobiLink [page 106]

Previous task: Lesson 13: Monitoring MobiLink Resources with the SQL Anywhere Monitor [page 125]

1.5.2 Tutorial: Using MobiLink with a SQL Anywhere Consolidated Database

This tutorial shows you how to mobilize a SQL Anywhere database by using MobiLink. It sets up synchronization between a SQL Anywhere consolidated database and an UltraLite remote database. You can also use a SQL Anywhere remote database.

Prerequisites

The following software is required:

- SQL Anywhere 17

You must have the following roles and privileges on the consolidated database:

- SYS_AUTH_DBA_ROLE compatibility role

Context

The purpose of this tutorial is to mobilize data for a mobile phone company that operates in many regions. In this scenario, each region:

- Is a remote synchronization environment.
- Has a local UltraLite database that is synchronized with the SQL Anywhere consolidated database at a central location, using MobiLink.
- Can access product information at its location and manipulate data from the remote database when a new customer activates an account or an existing customer activates a new mobile device.

This tutorial shows you how to:

- Evaluate important considerations, such as synchronization directions for remote tables, when designing a remote schema.
- Add unique primary keys to consolidated and remote databases.
- Set up synchronization between a consolidated database and a remote database using the *Create Synchronization Model Wizard*.
- Customize synchronization settings using SQL Central.
- Deploy a consolidated database and a remote database using the *Deploy Synchronization Model Wizard*.
- Synchronize the remote client with the consolidated database.

1. Designing the Schemas [page 131]
   This tutorial assumes that the sample database is installed on the computer where SQL Anywhere is running.
2. Lesson 1: Preparing the Consolidated Database [page 132]
   In this lesson you connect to the consolidated database, create the CustomerProducts table, and alter the Customers table to include regional information.
3. Lesson 2: Creating a Synchronization Model [page 134]
   In this lesson, you use the *Create Project Wizard* to create a new MobiLink project. The *Create Project Wizard* also creates a synchronization model using the defaults, which can be edited later.
4. Lesson 3: Deploying the Synchronization Model [page 137]
   The *Deploy Synchronization Model Wizard* allows you to deploy the consolidated database and remote database. You can deploy each database individually or both of them together. The *Deploy Synchronization Model Wizard* takes you through the steps of configuring options for deployment.
5. Lesson 4: Starting the MobiLink Server [page 139]
   In this lesson, you start the MobiLink server using the mlsrv17 -c option to connect to your consolidated database. You can use additional options to configure MobiLink server behavior.
6. Lesson 5: Synchronizing [page 140]
   In this lesson, you synchronize the MobiLink client with the MobiLink server using the ulsync utility to initiate synchronization.
7. Lesson 6: Cleaning Up [page 142]
   Remove the tutorials materials from your computer.

Related Information

MobiLink Synchronization [page 4]

1.5.2.1 Designing the Schemas

This tutorial assumes that the sample database is installed on the computer where SQL Anywhere is running.

The sample database is used as the consolidated database. The following table provides a description of each table in the SQL Anywhere consolidated database:

Table	Description
Customers	Customers whose information is kept on record.
SalesOrders	Records of account activations.
Products	Records of all products available for purchase.
CustomerProducts	A listing of the products each customer owns.

Designing the Remote Schema

It is unnecessary and inefficient for each region to have a copy of the entire consolidated database. The remote schema uses the same table names, but only contains information relevant to one particular region. To achieve this configuration, the remote schema is designed as a subset of the consolidated database in the following way:

Consolidated table	Remote table
Customers	Filter by *Region*.
SalesOrders	Filter by *Customer ID* for customers in the appropriate region.
Products	Include all rows.
CustomerProducts	Filter by *Customer ID* for customers in the appropriate region.

Each sales representative needs to keep information about the customers located in their region, as well as the products offered to all customers. However, a sales representative does not need information about customers in different regions, so this information is not synchronized to each regional office. This behavior is achieved by filtering rows based on a region identifier.

Note

You can also take a subset of columns from a table if certain columns are not required on the remote databases.

The next step is to choose the synchronization direction of each table. Consider what information a remote database needs to read and what information a remote database needs to create, change, or remove. In this example, a region needs access to the list of products offered to customers, but never enters a new product into the system. This creates the restriction that products must always enter the system from the consolidated database at the central location. However, a sales representative needs to record new account activations on a regular basis. These factors lead to the following synchronization directions for the tables:

Table	Synchronization
Customers	Upload to consolidated database only.
SalesOrders	Upload to consolidated database only.
Products	Download to remote database only.
CustomerProducts	Upload to consolidated database only.

Parent topic: Tutorial: Using MobiLink with a SQL Anywhere Consolidated Database [page 129]

Next task: Lesson 1: Preparing the Consolidated Database [page 132]

1.5.2.2 Lesson 1: Preparing the Consolidated Database

In this lesson you connect to the consolidated database, create the CustomerProducts table, and alter the Customers table to include regional information.

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

In a synchronization system, the primary key of a table is the only way to uniquely identify a row in different databases and the only way to detect conflicts. Every table that is being mobilized must have a primary key. The primary key must never be updated. You must also guarantee that a primary key value inserted at one database is not inserted at another database.

In a later lesson, the remote schema is created from the consolidated schema so the remote schema has the same primary keys as the consolidated schema.

Columns were specifically chosen to ensure unique primary keys for all databases. For the Customers table, the primary key consists of the ID column. Any value inserted into the remote Customers table must have a unique customer ID number (the Region value is always the same). This practice ensures uniqueness in each remote Customers table. The primary key in the consolidated Customers table prevents conflicts if multiple salespeople upload data. Each upload from a region is unique from another region because their Region values are different.

Procedure

1. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Central*.
2. Click *Connections* > *Connect With SQL Anywhere 17*.
3. Perform the following tasks in the *Connect* window:
   a. In the *Action* dropdown list, choose *Connect With An ODBC Data Source*.
   b. In the *User ID* field, type **DBA**.
   c. In the *Password* field, type **sql**.
   d. In the *ODBC Data Source name* field, type *SQL Anywhere 17 Demo*.
   e. Click *Connect*.
4. Connect to your consolidated database in Interactive SQL.

   At a command prompt, run the following command:

   ```
   dbisql -c "DSN=SQL Anywhere 17 Demo;UID=DBA;PWD=sql"
   ```
5. In Interactive SQL, execute the following statements to create and insert data in the CustomerProducts table:

   ```
   CREATE TABLE CustomerProducts
       (ID int default AUTOINCREMENT PRIMARY KEY,
       SalesOrderID int NOT NULL,
       CustomerID int NOT NULL,
       ProductID int);
   INSERT INTO CustomerProducts (SalesOrderID,CustomerID,ProductID)
       SELECT SalesOrders.ID, SalesOrders.CustomerID, SalesOrderItems.ProductID
       FROM SalesOrders, SalesOrderItems
       WHERE SalesOrders.ID = SalesOrderItems.ID;
   ```
6. In Interactive SQL, execute the following statements to add regional information for each customer to the Customers table:

   ```
   ALTER TABLE Customers
       ADD Region VARCHAR(255);
   UPDATE Customers
       SET Region = (SELECT TOP 1 SalesOrders.Region
       FROM SalesOrders
       WHERE Customers.ID = SalesOrders.CustomerID
       ORDER BY Region);
   COMMIT;
   ```

Results

A connection is made to the sample database, a table named CustomerProducts is created, and changes are made to the Customers table to include regional information.

Next Steps

Task overview: Tutorial: Using MobiLink with a SQL Anywhere Consolidated Database [page 129]

Previous: Designing the Schemas [page 131]

Next task: Lesson 2: Creating a Synchronization Model [page 134]

Related Information

MobiLink Consolidated Databases
Unique Primary Keys

1.5.2.3 Lesson 2: Creating a Synchronization Model

In this lesson, you use the *Create Project Wizard* to create a new MobiLink project. The *Create Project Wizard* also creates a synchronization model using the defaults, which can be edited later.

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 *SQL Central*, click *Tools* > *MobiLink 17* > *New Project*.
2. The *Create Project Wizard* appears.
3. In the *What do you want to name the new project* field, type **`mlsqla_project`**.
4. In the *Where do you want to save the new project* field, type `C:\mlsqla`, and click *Next*.
5. In the *Database display name* field, type *demo*.
6. Click *Edit*.
7. Perform the following tasks in the *Connect To A Generic ODBC Database* page:
    a. In the *User ID* field, type **`DBA`**.
    b. In the *Password* field, type **`sql`**.
    c. In the *ODBC Data Source name* field, click *Browse* and choose *SQL Anywhere 17 Demo*.
    d. Click *OK*, and click *Save*.
8. Select the *Remember the password* option, and click *Next*.

9. On the *New Remote Database Schema* page, in the *Which consolidated database tables and columns do you want to have in your remote database* list, select the following tables:
   - CustomerProducts
   - Customers
   - Products
   - SalesOrders

   Click *Next*.
10. Select the *Add a remote schema name to the project* option.
11. In the *What do you want to name the new remote schema* field, type **mlsqla_remote_schema** and click *Next*.
12. Choose *UltraLite* for the *Which type of remote database do you want to use* option and click *Finish*.
13. Click *Yes* when you are prompted to install the MobiLink setup scripts.
14. Click *Yes* if you are prompted to import the remote schema.
15. Click *OK*.
16. Right-click the new synchronization model and choose *Properties*.
    a. Type **sync_mlsqla** in the first field.
    b. Type **sync_mlsqla_publication** in the *Publication name* field.
    c. Type **sync_mlsqla_scriptversion** in the *Script version* field.

       The publication is the object on the remote database that specifies what data is synchronized. MobiLink server scripts define how uploaded data from remotes should be applied to the consolidated database, and how script versions group scripts. You can use different script versions for different applications, allowing you to maintain a single MobiLink server while synchronizing different applications.
    d. Click *Apply* and then click *OK*.
17. Perform the following tasks in the right pane of SQL Central:
    a. Click the *Events* tab.
    b. Update the CustomerProducts download_cursor to only download customer products for customers in the Eastern region.

       Replace the existing SQL script for the download_cursor event for the CustomerProducts table with the following query:

       ```
       SELECT "DBA"."CustomerProducts"."ID",
           "DBA"."CustomerProducts"."SalesOrderID",
           "DBA"."CustomerProducts"."CustomerID",
           "DBA"."CustomerProducts"."ProductID"
       FROM "DBA"."CustomerProducts"
       INNER JOIN "GROUPO"."Customers" ON "GROUPO"."Customers"."ID" =
           "DBA"."CustomerProducts"."CustomerID"
       WHERE "GROUPO"."Customers"."Region" = 'Eastern';
       ```
    c. Update the download cursor for the Customers table to only download customer information from the Eastern Region.

       Replace the existing SQL script for the download_cursor event for the Customers table with the following query:

       ```
       SELECT "GROUPO"."Customers"."ID",
           "GROUPO"."Customers"."Surname",
           "GROUPO"."Customers"."GivenName",
       ```

```
     "GROUPO"."Customers"."Street",
     "GROUPO"."Customers"."City",
     "GROUPO"."Customers"."State",
     "GROUPO"."Customers"."Country",
     "GROUPO"."Customers"."PostalCode",
     "GROUPO"."Customers"."Phone",
     "GROUPO"."Customers"."CompanyName",
     "GROUPO"."Customers"."Region"
FROM "GROUPO"."Customers"
WHERE Region = 'Eastern';
```

d. Update the SalesOrders download cursor to only download sales order information for customers in the Eastern region.

   Replace the existing SQL script for the download_cursor event for the SalesOrders table with the following query:

```
SELECT "GROUPO"."SalesOrders"."ID",
     "GROUPO"."SalesOrders"."CustomerID",
     "GROUPO"."SalesOrders"."OrderDate",
     "GROUPO"."SalesOrders"."FinancialCode",
     "GROUPO"."SalesOrders"."Region",
     "GROUPO"."SalesOrders"."SalesRepresentative"
FROM "GROUPO"."SalesOrders"
WHERE "GROUPO"."SalesOrders"."Region" = 'Eastern'
AND "GROUPO"."SalesOrders"."ID" IN
(SELECT "DBA"."CustomerProducts"."SalesOrderID"
FROM "DBA"."CustomerProducts");
```

18. Save the synchronization model. Click *File* > *Save*.

    The synchronization model is complete and ready for deployment.

Results

A MobiLink project and synchronization model are created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with a SQL Anywhere Consolidated Database [page 129]

Previous task: Lesson 1: Preparing the Consolidated Database [page 132]

Next task: Lesson 3: Deploying the Synchronization Model [page 137]

Related Information

Synchronization Models [page 37]
Consolidated Database Setup
MobiLink Server System Tables
MobiLink Server System Procedures
download_delete_cursor Scripts
Publications
Synchronization Model Tasks [page 41]
Table and Column Mappings [page 42]
Modifying the Download Strategy [page 47]

1.5.2.4 Lesson 3: Deploying the Synchronization Model

The *Deploy Synchronization Model Wizard* allows you to deploy the consolidated database and remote database. You can deploy each database individually or both of them together. The *Deploy Synchronization Model Wizard* takes you through the steps of configuring options for deployment.

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 under *MobiLink 17*, expand **`mlsqla_project`**, *Synchronization Models*, **`sync_mlsqla`**.
2. Click *File* > *Deploy*.

   The *Deploy Synchronization Model Wizard* appears. Select the default location for the generated files and click *Next*.
3. Accept the defaults on the *Client Network Options* page and click *Next*.
4. Perform the following tasks on the *MobiLink User And Password* page:
   a. Under *What MobiLink user and password do you want to use*, select *Use These*.
   b. In the *MobiLink user* field, type **`mlsqla_remote`**.
   c. In the *MobiLink password* field, type **`mlsqla_pass`**.
   d. Check *Register this user in the consolidated database. Registered users are permitted to synchronize*.
   e. Click *Next*.
5. Change the synchronization profile name to **`mlsqla_remote_syncprofile`** and click *Next*.

6. On the *Choose How To Prepare Databases For Synchronization* page, choose *Execute against consolidated database* for the *What do you want to do with the SQL script created to prepare the consolidated database for synchronization* option.
7. For the *What do you want to do with the SQL script created to prepare the remote database for synchronization* option, choose *Execute against a new remote database*.
8. In the *User ID* field, type **`DBA`**.
9. In the *Password* and *Retype Password* fields, type **`passwd`**. Click *Next*.
10. To review the SQL scripts created by the wizard, click *View*.
11. Click *Finish*.
12. Click *Close*.

Results

Your consolidated database is configured for synchronization with many remote clients, and you have successfully deployed one remote client. To deploy other remote clients, you can run this wizard again, making sure to create a new MobiLink user and opting out of deploying the consolidated database and MobiLink server. Since the consolidated and remote databases have already been deployed, just deploy other remote synchronization clients.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with a SQL Anywhere Consolidated Database [page 129]

Previous task: Lesson 2: Creating a Synchronization Model [page 134]

Next task: Lesson 4: Starting the MobiLink Server [page 139]

Related Information

Synchronization Model Deployment [page 58]
MobiLink Users in a Synchronization System
Using a SQL Anywhere Database as a Remote Database

1.5.2.5 Lesson 4: Starting the MobiLink Server

In this lesson, you start the MobiLink server using the mlsrv17 -c option to connect to your consolidated database. You can use additional options to configure MobiLink server behavior.

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. At a command prompt, change to the `c:\mlsqla` directory.
2. Connect to your consolidated database by running the following command:

   ```
   mlsrv17 -c "DSN=SQL Anywhere 17 Demo;UID=DBA;PWD=sql" -o mlsrv.mls -v+ -dl -
   zf -zu+ -x tcpip
   ```

   Below is a description of each MobiLink server option used in this tutorial. The options -o, -v, and -dl provide debugging and troubleshooting information. Using these logging options is appropriate in a development environment. For performance reasons, -v+ and -dl are typically not used in production environments.

Option	Description
-c	Precedes the connection string.
-o	Specifies the message log file `mlsrv.mls`.
-v+	Specifies what information is logged. Using -v+ sets maximum verbose logging.
-dl	Displays all log messages on screen.
-zf	Causes the MobiLink server to check for script changes at the beginning of each synchronization.
-zu+	Adds new users automatically.
-x	Sets the communications protocol and parameters for MobiLink clients.

   > **i Note**
   >
   > The -zf and -zu+ options should be used for debugging and development purposes only. This tutorial requires the -zf option so that you do not need to shut down the server when adding new scripts to the consolidated database in a later lesson. The -zu+ option automatically adds new MobiLink users to the synchronization environment.

The MobiLink server messages window appears.

Results

The MobiLink server is started and connected to the consolidated database.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with a SQL Anywhere Consolidated Database [page 129]

Previous task: Lesson 3: Deploying the Synchronization Model [page 137]

Next task: Lesson 5: Synchronizing [page 140]

Related Information

MobiLink Server Options

1.5.2.6 Lesson 5: Synchronizing

In this lesson, you synchronize the MobiLink client with the MobiLink server using the ulsync utility to initiate synchronization.

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. Change to the directory `c:\mlsqla\mlsqla_project\sync_mlsqla_deploy` and run the following command to synchronize the **`sync_mlsqla_remote`** database:

   ```
   ulsync -c "DBF=sync_mlsqla_remote.udb"
   "Publications=sync_mlsqla_publication;MobiLinkUid=mlsqla_remote;MobiLinkPwd=ml
   sqla_pass;ScriptVersion=sync_mlsqla_scriptversion;Stream=tcpip{port=2439}"
   ```

   DBF

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

   Publications

   is the publication on the remote device that is used to perform the synchronization. (This publication was created by the *Create Synchronization Model Wizard*.)

   MobiLinkUid

   is the user name used to authenticate with the MobiLink server.

   MobiLinkPwd

   is the password used to authenticate with the MobiLink server.

   ScriptVersion

   is the script version on the remote device that is used to perform the synchronization. (This publication was created by the *Create Synchronization Model Wizard*.)

   Stream

   sets options to configure the network protocol.

   The progress of the synchronization appears in the MobiLink server messages window. When this command runs successfully, the ulsync application populates the remote database with a subset of information from the consolidated database.

   If synchronization fails, check the connection information you passed to the ulsync application, and the MobiLink user name and password. Failing that, check the publication name you used, and ensure that the consolidated database and MobiLink server are running. You can also examine the contents of the synchronization logs (server and client).

   > i Note
   >
   > If you are running the ulsync application on a different computer from your MobiLink server, you must pass in arguments that specify the location of the MobiLink server.

   After successfully synchronizing the remote client to the consolidated database through the MobiLink server, the remote database should be populated with information relevant to one region. You can verify that the database is populated in SQL Central using the SQL Anywhere 17 plug-in.

2. Open SQL Central.
3. Connect to the remote database:
   a. In the left pane, right-click *UltraLite 17*, and click *Connect*.
   b. Type **`DBA`** as the *User ID* and **`passwd`** as the *Password*.
   c. In the *Database File* field, type **`C:\mlsqla\mlsqla_project\sync_mlsqla_deploy\sync_mlsqla_remote.udb`**

   d. Click *Connect*.
4. In the left pane, expand *UltraLite 17*, **`sync_mlsqla_remote`**, *Tables*, *Customers*.
5. Click the *Data* tab in the right pane.

   In the Customers tables, all the records are for the customers pertaining to the Eastern region. This particular region is not concerned with the customer information of other regions. For this reason, you set the synchronization scripts to filter out rows by region, and you set this database's remote ID to the value of a particular region identifier. This particular region's database takes up less space, and requires less time to synchronize. Since the remote database size is kept to a minimum, frequently performed operations such as entering a new customer or processing a change in a mobile device run faster and more efficiently.

Results

Data is synchronized between the remote and consolidated databases.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with a SQL Anywhere Consolidated Database [page 129]

Previous task: Lesson 4: Starting the MobiLink Server [page 139]

Next task: Lesson 6: Cleaning Up [page 142]

Related Information

UltraLite Synchronization Utility (ulsync)

1.5.2.7 Lesson 6: Cleaning Up

Remove the tutorials materials from your computer.

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. Close all instances of the following applications:
   - SQL Central
   - Interactive SQL
2. Delete the `C:\mlsqla` directory containing your consolidated and remote databases.
3. Run the following command to erase the sample database and create a new copy of the sample database with its original objects and data:

   ```
   newdemo "%SQLANYSAMP17%\demo.db"
   ```

   When you are prompted, choose to erase any existing files.

Results

The tutorial materials are removed from your computer.

Task overview: Tutorial: Using MobiLink with a SQL Anywhere Consolidated Database [page 129]

Previous task: Lesson 5: Synchronizing [page 140]

1.5.3 Tutorial: Using MobiLink with an Oracle Database 11g

This tutorial shows you to mobilize an Oracle Database 10g using MobiLink. It sets up synchronization between an Oracle Database 11g and a SQL Anywhere remote database. You could also set up an UltraLite remote database.

Prerequisites

The following software is required:

- SQL Anywhere 17
- Oracle Database 11g Release 2 or later

You must have the following permissions on the Oracle database:

- SELECT from SYS.GV_$TRANSACTION
- SELECT from SYS.GV_$SESSION

- SELECT from SYS.V_$SESSION
- SELECT from SYS.GV_$LOCK
- EXECUTE on SYS.DBMS_UTILITY
- SELECT from DBA_OBJECTS

You must have the following roles and privileges on the remote database:

- SYS_REPLICATION_ADMIN_ROLE system role
- SYS_RUN_REPLICATION_ROLE system role

Context

The purpose of this tutorial is to mobilize the data pertaining to a sales team. In this scenario, each salesperson is a remote synchronization client. Each salesperson has a local SQL Anywhere database that is synchronized to a corporate Oracle database at headquarters using MobiLink. Each salesperson accesses corporate data with their laptop or mobile device, and manipulates data from the remote database.

This tutorial assumes you performed a basic installation of Oracle Database 11g, which creates a starter database named orcl. The orcl database has the Order Entry (OE) and Human Relations (HR) sample schemas. Alternatively, you can create a new database using the Oracle Database Configuration Assistant and install the sample schemas or manually install the sample schemas into a blank database via SQL*Plus. The sample schema SQL files are available through separate **Example** download packages directly from Oracle.

This tutorial assumes that you can connect to Oracle as the SYS user with SYSDBA privileges. This is a requirement when you grant permission for the GV_$TRANSACTION Oracle system view. The password for the SYS user is set during installation of an Oracle database.

This tutorial shows you how to:

- Evaluate important considerations, such as synchronization directions for remote tables, when designing a remote schema.
- Add unique primary keys to consolidated and remote databases.
- Create an ODBC data source that connects MobiLink to an Oracle Database 11g.
- Set up synchronization between a consolidated database and remote database using the *Create Synchronization Model Wizard*.
- Customize a synchronization model using SQL Central.
- Deploy a consolidated database and remote database using the *Deploy Synchronization Model Wizard*.
- Synchronize the remote client with the consolidated database.

1. Designing the Schemas [page 146]
   This tutorial assumes that you have installed the Order Entry (OE) and Human Relations (HR) sample schemas. The OE schema is used as the consolidated database. It encapsulates information about employees, orders, customers, and products. For this tutorial, you are primarily interested in the OE schema.
2. Lesson 1: Preparing the Consolidated Database [page 147]
   The OE database needs to be altered for use with MobiLink. Columns are dropped because they were created as user-defined types. You must also grant the OE user the privilege to create triggers because MobiLink needs to create triggers using OE's credentials.

3. Lesson 2: Adding Unique Keys to the Consolidated Database [page 149]
   In a synchronization system, the primary key of a table is the only way to uniquely identify a row in different databases and the only way to detect conflicts.
4. Lesson 3: Connecting with MobiLink [page 150]
   In this lesson, you create an ODBC data source that connects MobiLink to the consolidated database.
5. Lesson 4: Creating a MobiLink Project and Synchronization Model [page 152]
   In this lesson, you connect to the consolidated database by creating a new MobiLink project. A synchronization model is created automatically when you create a project.
6. Lesson 5: Modifying a Synchronization Model [page 153]
   In this lesson, you modify the synchronization model for your consolidated database that was created when you created a new MobiLink project.
7. Lesson 6: Deploying the Synchronization Model [page 155]
   The *Deploy Synchronization Model Wizard* allows you to deploy the consolidated database and remote database. You can deploy each database individually or both of them together. The *Deploy Synchronization Model Wizard* takes you through the steps of configuring options for deployment.
8. Lesson 7: Starting the Server And Client [page 157]
   In a previous lesson, you modified the download cursor script to download information related to one salesperson. In this lesson, you specify the salesperson by setting the remote ID to the salesperson identifier, and start the MobiLink consolidated and remote database.
9. Lesson 8: Setting the Remote ID [page 159]
   In this lesson, you set the database's remote ID to the value of a valid salesperson identifier.
10. Lesson 9: Synchronizing the Remote Client [page 161]
   In this lesson, you synchronize the remote client using the dbmlsync utility. Dbmlsync connects to the remote database, authenticates itself with the MobiLink server, and performs all the uploads and downloads necessary to synchronize the remote and consolidated databases based on a publication in the remote database.
11. Lesson 10: Viewing the Data in the Remote Database [page 163]
   After successfully synchronizing the remote client to the consolidated database through the MobiLink server, the remote database should be populated with information relevant to one salesperson. You can verify that the database is populated correctly in SQL Central using the SQL Anywhere 17 plug-in.
12. Lesson 11: Cleaning Up [page 164]
   Regenerate the Order Entry database and remove all tutorial materials from your computer.

Related Information

MobiLink Synchronization [page 4]
MobiLink Plug-in for SQL Central [page 27]
Oracle sample schemas

1.5.3.1 Designing the Schemas

This tutorial assumes that you have installed the Order Entry (OE) and Human Relations (HR) sample schemas. The OE schema is used as the consolidated database. It encapsulates information about employees, orders, customers, and products. For this tutorial, you are primarily interested in the OE schema.

Here is a brief description of the relevant tables in the OE schema:

Table	Description
CUSTOMERS	Customers whose information is kept on record.
INVENTORIES	How much of each product is stored in each warehouse.
ORDER_ITEMS	A list of products included in each order.
ORDERS	A record of a sale between a salesperson and a customer on a specific date.
PRODUCT_DESCRIPTIONS	Descriptions of each product in different languages.
PRODUCT_INFORMATION	A record of each product in the system.

Designing the Remote Schema

It is unnecessary and inefficient for each salesperson to have a copy of the entire consolidated database. The remote schema is designed so that it only contains information relevant to one particular salesperson. To achieve this, the remote schema is designed in the following way:

Consolidated table	Remote table
CUSTOMERS	Includes all rows.
INVENTORIES	Not included on remote.
ORDER_ITEMS	Filter by sales_rep_id.
ORDERS	Includes all rows.
PRODUCT_DESCRIPTIONS	Not included on remote.
PRODUCT_INFORMATION	Includes all rows.

Each salesperson needs to keep records of all customers and products, so that any product can be sold to any customer. This tutorial assumes that a salesperson always speaks the same language as the customer, so you do not need the PRODUCT_DESCRIPTIONS table. Each salesperson needs information about orders, but not orders related to other salespeople. This is achieved by filtering rows based on salesperson identifier.

The next step is to choose the synchronization direction of each table. You should consider what information a remote database needs to read and what information a remote database needs to create, change, or remove. In this example, a specific salesperson needs a list of products and customers, but never enters a new product into the system. You are making the restriction that products and customers always enter the system from the consolidated database at headquarters. However, a salesperson needs to record new orders on a regular basis. These factors lead to the following decisions about the synchronization in each table:

Table	Synchronization
CUSTOMERS	Download to remote database only.
ORDER_ITEMS	Download and upload.
ORDER	Download and upload.
PRODUCT_INFORMATION	Download to remote database only.

Parent topic: Tutorial: Using MobiLink with an Oracle Database 11g [page 143]

Next task: Lesson 1: Preparing the Consolidated Database [page 147]

1.5.3.2 Lesson 1: Preparing the Consolidated Database

The OE database needs to be altered for use with MobiLink. Columns are dropped because they were created as user-defined types. You must also grant the OE user the privilege to create triggers because MobiLink needs to create triggers using OE's credentials.

Prerequisites

This tutorial assumes that you have installed the Order Entry (OE) sample database.

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

Information about installing the sample schema for Oracle 11g can be found in Oracle documentation.

Columns created as user-defined types could be translated into types that SQL Anywhere recognizes, but doing so is not relevant to this tutorial.

Procedure

1. Connect as the SYS user with SYSDBA privileges using the Oracle SQL Plus application. At a command prompt, run the following command:

```
sqlplus SYS/your password for sys as SYSDBA
```

2. To drop columns created as user-defined types, execute the following statements:

```
ALTER TABLE OE.CUSTOMERS DROP COLUMN CUST_ADDRESS;
ALTER TABLE OE.CUSTOMERS DROP COLUMN PHONE_NUMBERS;
ALTER TABLE OE.CUSTOMERS DROP COLUMN CUST_GEO_LOCATION;
ALTER TABLE OE.PRODUCT_INFORMATION DROP COLUMN WARRANTY_PERIOD;
```

3. To unlock the OE user and set the password to sql, execute the following statement:

```
ALTER USER OE IDENTIFIED BY sql ACCOUNT UNLOCK;
```

4. To allow the OE user to create triggers, execute the following statement:

```
GRANT CREATE ANY TRIGGER TO OE;
```

5. To drop the orders_customer foreign key and create a new foreign key that references the customer_id in the customers table, run the following commands:

```
ALTER TABLE OE.ORDERS DROP CONSTRAINT ORDERS_CUSTOMER_ID_FK;
ALTER TABLE OE.ORDERS ADD CONSTRAINT ORDERS_CUSTOMER_ID_FK
  FOREIGN KEY (CUSTOMER_ID) REFERENCES OE.CUSTOMERS (CUSTOMER_ID);
```

Results

Columns created as user-defined types are dropped and the OE user is now able to create triggers.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Oracle Database 11g [page 143]

Previous: Designing the Schemas [page 146]

Next task: Lesson 2: Adding Unique Keys to the Consolidated Database [page 149]

Related Information

MobiLink Consolidated Databases
Oracle Consolidated Database
Unique Primary Keys
Oracle Database 10g and 11g documentation

1.5.3.3 Lesson 2: Adding Unique Keys to the Consolidated Database

In a synchronization system, the primary key of a table is the only way to uniquely identify a row in different databases and the only way to detect conflicts.

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

Every table that is being mobilized must have a primary key. The primary key must never be updated. You must also guarantee that a primary key value inserted at one database is not inserted in another database.

There are several ways of generating unique primary keys. For simplicity, the method of composite primary keys is used in this tutorial. This method creates primary keys with multiple columns that are unique across the consolidated and remote databases.

Procedure

1. At a command prompt, run the following command:

```
sqlplus SYS/your password for sys as SYSDBA
```

2. Values added to the SALES_REP_ID must exist in the HR.EMPLOYEES table. The ORDERS_SALES_REP_FK foreign key enforces this rule. Execute the following statement to drop the foreign key:

```
ALTER TABLE OE.ORDERS
DROP CONSTRAINT ORDERS_SALES_REP_FK;
```

3. The SALES_REP_ID column cannot be added as a primary key because it contains null values. For this tutorial, replace the null values with a value of 1. Execute the following statement:

```
UPDATE OE.ORDERS
SET SALES_REP_ID = 1
WHERE SALES_REP_ID IS NULL;
```

4. The ORDER_ID column is the current primary key of the ORDERS table. To drop the current primary key, execute the following statement:

```
ALTER TABLE OE.ORDERS
DROP PRIMARY KEY CASCADE;
```

5. The composite primary key consists of the SALES_REP_ID column and the ORDER_ID column. To add the composite primary key, execute the following statement:

```
ALTER TABLE OE.ORDERS
ADD CONSTRAINT salesrep_order_pk PRIMARY KEY (sales_rep_id, order_id);
```

Results

After executing these statements, the MobiLink server connects to the consolidated database and sets up synchronization for any number of remote databases.

In a later lesson, the remote schema is created from the consolidated schema so the remote schema has the same primary keys as the consolidated schema.

Columns were specifically chosen to ensure unique primary keys for all databases. For the ORDERS table, the primary key consists of the SALES_REP_ID and ORDER_ID columns. Any value inserted into the remote sales table must have an unique order number (the SALES_REP_ID value is always the same). This practice ensures uniqueness in each remote ORDERS table. The primary key in the consolidated ORDERS table prevents conflicts if multiple salespeople upload data. Each upload from a salesperson is unique to another salesperson because their SALES_REP_ID values are different.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Oracle Database 11g [page 143]

Previous task: Lesson 1: Preparing the Consolidated Database [page 147]

Next task: Lesson 3: Connecting with MobiLink [page 150]

1.5.3.4 Lesson 3: Connecting with MobiLink

In this lesson, you create an ODBC data source that connects MobiLink to the consolidated 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. Create an ODBC data source.

   You should use the SQL Anywhere 17 - Oracle ODBC driver that comes with SQL Anywhere 17. Use the following configuration settings:

ODBC tab fields	Values
Data Source Name	oracle_cons
User ID	OE
Password	passwd
TNS Service Name	orcl
Procedure Returns Results Or Uses VARRAY Parameters	selected
Array Size	60000

   This tutorial assumes you performed a basic installation of Oracle Database 11g, which creates a starter database named orcl. The Order Entry (OE) schema is automatically installed on orcl. If you installed the OE schema on another database, use the name of the database as the TNS service name value.

2. Click *Test Connection* to test the ODBC connection.

Results

The ODBC data source is created and tested.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Oracle Database 11g [page 143]

Previous task: Lesson 2: Adding Unique Keys to the Consolidated Database [page 149]

Next task: Lesson 4: Creating a MobiLink Project and Synchronization Model [page 152]

Related Information

SQL Anywhere 17 - Oracle ODBC Driver
Recommended ODBC Drivers For MobiLink

1.5.3.5 Lesson 4: Creating a MobiLink Project and Synchronization Model

In this lesson, you connect to the consolidated database by creating a new MobiLink project. A synchronization model is created automatically when you create a project.

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. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Central*.
2. Click *Tools* > *MobiLink 17* > *New Project*.

   The *Create Project Wizard* appears.
3. In the *What do you want to name the new project* field, type **`oracle_project`**.
4. In the *Where do you want to save the new project* field, type `C:\mlora`, and click *Next*.
5. In the *Database display name* field, type **`oracle_cons`**.
6. Click *Edit*.
7. Perform the following tasks in the *Connect To A Generic ODBC Database* page:
   a. In the *User ID* field, type **`OE`**.
   b. In the *Password* field, type the password for the **`passwd`** account.
   c. In the *ODBC Data Source name* field, click *Browse*, and choose **`oracle_cons`**.
   d. Click *OK*, and click *Save*.
8. Select the *Remember the password* option, and click *Next*.
9. In the *Consolidated Schema Owners* window, select *Only load the database schema for selected owners* and check *OE user*.
10. On the *New Remote Database Schema* page, in the *Which consolidated database tables and columns do you want to have in your remote database* list, choose the following tables and then click *Next*:
    - CUSTOMERS
    - ORDERS
    - ORDER_ITEMS
    - PRODUCT_INFORMATION
11. Choose the *Add a remote schema name to the project* option.
12. Type **`oracle_remote_schema`** for the remote schema name, and then click *Next*.
13. Select the *SQL Anywhere* option, and then click *Finish*.

If this is the first time the consolidated database has been used by MobiLink, a message appears asking you to install the MobiLink system setup. Installing the MobiLink system setup adds MobiLink system tables and procedures. Click *Yes*, and then click *OK*.

Results

A MobiLink project and synchronization model are created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Oracle Database 11g [page 143]

Previous task: Lesson 3: Connecting with MobiLink [page 150]

Next task: Lesson 5: Modifying a Synchronization Model [page 153]

1.5.3.6 Lesson 5: Modifying a Synchronization Model

In this lesson, you modify the synchronization model for your consolidated database that was created when you created a new MobiLink project.

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. Right click the *oracle_remote_schema* synchronization model and choose *Properties*.
2. Type `sync_oracle` in the first field.
3. Perform the following tasks:
    a. In the *Publication name* field, type `sync_oracle_publication`.
    b. In the *Script version* field, type `sync_oracle_scriptversion`.

The publication is the object on the remote database that specifies what data is synchronized. MobiLink server scripts define how uploaded data from remotes should be applied to the consolidated database, and script versions group scripts. You can use different script versions for different applications, allowing you to maintain a single MobiLink server while synchronizing different applications.

  c. Click *Apply* and then click *OK*.

4. Set the direction that data is synchronized for each table in the synchronization model.

   Click the *Mappings* tab in the right pane, and set the rows in the *Mapping Direction* column as follows:

   - The **ORDERS** and **ORDER_ITEMS** tables should be set to *Bi-directional* (both upload and download).
   - The remaining tables should be set to *Download to remote only*.

5. If a window appears indicating that loading the consolidated schema for all owners may take a long time, choose to load the database schema for the **HR** and **OE** users.
6. Filter the rows downloaded to the remote database by remote ID.
   a. Select the row containing the **ORDERS** table and then Click the *Download Subset* tab at the bottom of the right pane.
   b. Change the *Download Subset* column to *Custom*.
   c. Filter the rows by remote ID, which uniquely identifies the remote database, by adding a restriction to the WHERE clause of the download_cursor script.

      Type a search condition in the *SQL expression to use in the download cursor's WHERE clause* field. For example, the following SQL script can be used for the **ORDERS** table:

      ```
      OE.ORDERS.SALES_REP_ID = {ml s.remote_id}
      ```

      The download cursor script specifies what columns and rows are downloaded from each table to the remote database. The search condition ensures that you only download information about one sales representative, namely, the sales representative that has an identifier that equals the remote ID for the database.
   d. Click the *Download Delete Subset* tab and change the *Download Delete Subset* from *Downloaded* to *All*.
7. Save the synchronization model.

   Click *File* > *Save*.

Results

The synchronization model is complete and ready for deployment.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Oracle Database 11g [page 143]

Previous task: Lesson 4: Creating a MobiLink Project and Synchronization Model [page 152]

Next task: Lesson 6: Deploying the Synchronization Model [page 155]

Related Information

Consolidated Database Setup
MobiLink Server System Tables
MobiLink Server System Procedures
Synchronization Scripts
download_delete_cursor Scripts
Conflict Handling Overview
Conflict Resolution
Publications
Synchronization Events
Synchronization Model Tasks [page 41]
Synchronization Techniques
Table and Column Mappings [page 42]
Modifying the Download Strategy [page 47]
Modifying Conflict Detection and Resolution [page 55]

1.5.3.7 Lesson 6: Deploying the Synchronization Model

The *Deploy Synchronization Model Wizard* allows you to deploy the consolidated database and remote database. You can deploy each database individually or both of them together. The *Deploy Synchronization Model Wizard* takes you through the steps of configuring options for deployment.

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 under *MobiLink 17*, expand **`oracle_project`**, *Synchronization Models*, **`sync_oracle`**.
2. Click *File* > *Deploy*.

   The *Deploy Synchronization Model Wizard* appears.

3. Accept the default setting for the *Select The Folder To Contain Files Generated By The Wizard* field and click *Next*.
4. Accept the default settings for the *Client Network Options* page and click *Next*.
5. On the *MobiLink User and Password* page under *What MobiLink user and password do you want to use*, select *Use these* and perform the following tasks:
    a. In the *MobiLink User* field, type **`oracle_remote`**.
    b. In the *MobiLink Password* field, type **`oracle_pass`**.
    c. Check *Register this user in the consolidated database. Registered users are permitted to synchronize* and click *Next* and then click *Next* again.
6. On the *Synchronization Profile* page, leave the default synchronization profile name as **`sync_oracle_publication_oracle_remote`**. Click *Next*.
7. On the *Choose How To Prepare Databases For Synchronization* page, choose the following options:
    a. For *What do you want to do with the SQL script created to prepare the consolidated database for synchronization*, choose *Execute against consolidated database*.
    b. For *What do you want to do with the SQL script created to prepare the remote database for synchronization*, choose *Execute against a new remote database*.
    c. In the *User ID* field, type **`remoteuser`**.
    d. In the *Password* and *Retype Password* fields, type **`remotepass`** and click *Next*.
8. Click *Close* if the *Generating Deployment Files* window appears.
9. Click *Finish*.
10. Click *Close*.

Results

Your consolidated database is configured for synchronization with many remote clients, and you have successfully deployed one remote client. To deploy other remote clients, you can run this wizard again, making sure to create a new MobiLink user and opt out of deploying the consolidated database and MobiLink server. Since the consolidated database and MobiLink server have already been deployed, all you need to do is deploy other remote synchronization clients.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Oracle Database 11g [page 143]

Previous task: Lesson 5: Modifying a Synchronization Model [page 153]

Next task: Lesson 7: Starting the Server And Client [page 157]

Related Information

Synchronization Model Deployment [page 58]
MobiLink Users in a Synchronization System
Using a SQL Anywhere Database as a Remote Database

1.5.3.8 Lesson 7: Starting the Server And Client

In a previous lesson, you modified the download cursor script to download information related to one salesperson. In this lesson, you specify the salesperson by setting the remote ID to the salesperson identifier, and start the MobiLink consolidated and remote 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

By default, MobiLink uses the snapshot/READ COMMITTED isolation level for upload and download. For the MobiLink server to make the most effective use of snapshot isolation, the Oracle account used by the MobiLink server must have access to the GV_$TRANSACTION Oracle system view. If access is not given, a warning is issued and rows may be missed on download.

Procedure

1. Connect as the SYS user with SYSDBA privileges using the Oracle SQL Plus application. At a command prompt, run the following command:

   ```
   sqlplus SYS/your password for sys as SYSDBA
   ```
2. To grant access to the GV_$TRANSACTION Oracle system view, execute the following statement:

   ```
   GRANT SELECT ON SYS.GV_$TRANSACTION TO OE;
   ```
3. To grant access to the V$SESSION and GV_$SESSION Oracle system views, execute the following statement:

   ```
   GRANT SELECT ON SYS.V_$SESSION TO OE;
         GRANT SELECT ON SYS.GV_$SESSION TO OE;
   ```

4. To grant access to other system objects, execute the following statement:

```
GRANT SELECT ON SYS.GV_$LOCK TO OE;
GRANT EXECUTE ON SYS.DBMS_UTILITY TO OE;
GRANT SELECT ON DBA_OBJECTS TO OE;
```

5. At a command prompt, navigate to the directory where you created the synchronization model. (This is the root directory you chose in the first step of the *Create Synchronization Model Wizard*.)

   If you used the suggested directory names, navigate to the following directory: `mlora\oracle_project \sync_oracle_deploy\`.
6. To start the MobiLink server, run the following command:

```
mlsrv.bat "DSN=oracle_cons;UID=OE;PWD=passwd"
```

   mlsrv.bat

   is the command file created to start the MobiLink server.

   DSN

   is the ODBC data source name.

   UID

   is the user name you use to connect to the consolidated database.

   PWD

   is the password you use to connect to the consolidated database.

   If the MobiLink server fails to start, check the connection information for the consolidated database.
7. At a command prompt, navigate to the directory where the *Deploy Synchronization Model Wizard* created your remote database.

   If you used the suggested directory names, navigate to the following directory: `mlora\oracle_project \sync_oracle_deploy\`.
8. Start your remote SQL Anywhere database by running the following command:

```
dbsrv17 -n remote_eng sync_oracle_remote.db -n remote_db
```

   dbsrv17

   is the database server used to start the SQL Anywhere database.

   remote_eng

   is the database server name.

   sync_oracle_remote.db

   is the database file that is started on remote_eng.

   remote_db

   is the name of the database on remote_eng.

Results

When this command runs successfully, a SQL Anywhere database server named remote_eng starts and loads the database called remote_db.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Oracle Database 11g [page 143]

Previous task: Lesson 6: Deploying the Synchronization Model [page 155]

Next task: Lesson 8: Setting the Remote ID [page 159]

Related Information

Deployed Synchronization Models [page 66]
MobiLink Server
Remote IDs
SQL Anywhere Database Server Executable (dbsrv17, dbeng17)

1.5.3.9 Lesson 8: Setting the Remote ID

In this lesson, you set the database's remote ID to the value of a valid salesperson identifier.

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

In the remote schema, each remote database represents one salesperson. The synchronization scripts you wrote included logic that instructed the MobiLink server to download a subset of data based on the remote ID of the remote database. You must set the database's remote ID to the value of a valid salesperson identifier before the first synchronization because when the remote device synchronizes for the first time, it downloads all information related to the chosen salesperson.

Procedure

1. Choose a valid salesperson identifier:
    a. Connect as the SYS user with SYSDBA privileges using the Oracle SQL Plus application. At a command prompt, run the following command:

    ```
    sqlplus SYS/your-password-for-sys as SYSDBA
    ```

    b. To view a list of valid salesperson identifiers in the ORDERS table, execute the following statement:

    ```
    SELECT COUNT( SALES_REP_ID ), SALES_REP_ID
    FROM OE.ORDERS GROUP BY SALES_REP_ID;
    ```

    In this example, the remote database represents a salesperson with a SALES_REP_ID of 154.

    c. To exit Oracle SQL Plus, run the following command:

    ```
    exit
    ```

2. To set the database's remote ID to a value of 154, run the following command:

    ```
    dbisql -c "SERVER=remote_eng;DBN=remote_db;UID=DBA;PWD=passwd" "SET OPTION
    PUBLIC.ml_remote_id='154'"
    ```

    dbisql

    is the application used to execute SQL commands against a SQL Anywhere database.

    ENG

    specifies the database server name remote_eng.

    DBN

    specifies the database name remote_db.

    UID

    is the user name used to connect to your remote database.

    PWD

    is the password used to connect to your remote database.

    SET OPTION PUBLIC.ml_remote_id='154'

    is the SQL statement used to set the remote ID to a value of 154.

Results

The database's remote ID is set to the value of a valid salesperson identifier.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Oracle Database 11g [page 143]

Previous task: Lesson 7: Starting the Server And Client [page 157]

Next task: Lesson 9: Synchronizing the Remote Client [page 161]

1.5.3.10 Lesson 9: Synchronizing the Remote Client

In this lesson, you synchronize the remote client using the dbmlsync utility. Dbmlsync connects to the remote database, authenticates itself with the MobiLink server, and performs all the uploads and downloads necessary to synchronize the remote and consolidated databases based on a publication in the remote 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

At a command prompt, run the following command:

```
dbmlsync
-c "SERVER=remote_eng;DBN=remote_db;UID=DBA;PWD=sql"
-n sync_oracle_publication
-u oracle_remote -mp oracle_pass
```

dbmlsync

is the synchronization application.

SERVER

specifies the name of the remote database server.

DBN

specifies the name of the remote database.

UID

specifies the user name used to connect to the remote database.

PWD

specifies the password used to connect to the remote database.

sync_oracle_publication

is the publication on the remote device that is used to perform the synchronization. (This publication was created by the *Create Synchronization Model Wizard*.)

oracle_remote

is the user name used to authenticate with the MobiLink server.

oracle_pass

is the password used to authenticate with the MobiLink server.

i Note

If you are running the dbmlsync application on a different computer from your MobiLink server, you must pass in arguments that specify the location of the MobiLink server.

Results

The progress of the synchronization appears in the *SQL Anywhere MobiLink Client Messages* window. When this command runs successfully, the dbmlsync application populates the remote database with a subset of information from the consolidated database.

If synchronization fails, check the connection information you pass to the dbmlsync application, and the MobiLink user name and password. Failing that, check the publication name you used, and ensure that the consolidated database and MobiLink server are running. You can also examine the contents of the synchronization logs (server and client).

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Oracle Database 11g [page 143]

Previous task: Lesson 8: Setting the Remote ID [page 159]

Next task: Lesson 10: Viewing the Data in the Remote Database [page 163]

Related Information

The Synchronization Process [page 18]
MobiLink SQL Anywhere Client Utility (dbmlsync) Syntax

1.5.3.11 Lesson 10: Viewing the Data in the Remote Database

After successfully synchronizing the remote client to the consolidated database through the MobiLink server, the remote database should be populated with information relevant to one salesperson. You can verify that the database is populated correctly in SQL Central using the SQL Anywhere 17 plug-in.

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 SQL Central.
2. Connect to the remote database:
    a. In the left pane, right-click *SQL Anywhere 17*, and click *Connect*.
    b. In the *Authentication* dropdown list, click *Database*, and type **`DBA`** as the *User ID* and **`passwd`** as the *Password*.
    c. In the *Action* dropdown list, click *Connect to a running database on this computer*. Type **`remote_eng`** as the *Server name* and **`remote_db`** as the *Database name*.
    d. Click *Connect*.
3. In the left pane of SQL Central under *remote_db - DBA*, expand *Tables*, click the ORDERS table, and then click the *Data* tab in the right pane.

   In the ORDERS tables, all the records are for the salesperson with an identifier of 154. This particular salesperson is not concerned with the sales information of other salespeople. For this reason, you set the synchronization scripts to filter out rows by the remote ID, and you set this database's remote ID to the value of a particular salesperson identifier. Now this particular salesperson's database takes up less space, and requires less time to synchronize. Since the remote database size is kept to a minimum, frequently performed operations, such as entering a new sale or processing a refund on a previous sale, run faster and more efficiently.

Results

The remote database is populated only with information relevant to the salesperson with an identifier of 154.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Oracle Database 11g [page 143]

Previous task: Lesson 9: Synchronizing the Remote Client [page 161]

Next task: Lesson 11: Cleaning Up [page 164]

1.5.3.12 Lesson 11: Cleaning Up

Regenerate the Order Entry database and remove all tutorial materials from your computer.

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. Follow the steps described in Oracle sample database to recreate the Oracle sample database.
2. Delete the MobiLink project.
   a. Start SQL Central.
   b. In the right pane, double-click *MobiLink 17*.
   c. The **`oracle_project`** appears in the right pane.
   d. Right click **`oracle_project`**, and click *Delete*.
   e. In the *Delete Project* window, choose *Remove from list and my computer* and click *Yes*.

Results

The Order Entry database is regenerated and all tutorial materials are removed from your computer.

Task overview: Tutorial: Using MobiLink with an Oracle Database 11g [page 143]

Previous task: Lesson 10: Viewing the Data in the Remote Database [page 163]

1.5.4 Tutorial: Using MobiLink with an Adaptive Server Enterprise Consolidated Database

This tutorial shows you how to mobilize an Adaptive Server Enterprise database using MobiLink. It sets up synchronization between an Adaptive Server Enterprise consolidated database and a SQL Anywhere remote database. You could also use UltraLite clients.

Prerequisites

The following software is required:

- SQL Anywhere 17
- Adaptive Server Enterprise 15.7

You must have the following roles and privileges on the consolidated database:

- SELECT permission on MASTER..SYSTRANSACTIONS and MASTER..SYSPROCESSES

You must have the following roles and privileges on the SQL Anywhere remote:

- SYS_REPLICATION_ADMIN_ROLE system role
- SYS_RUN_REPLICATION_ROLE system role

Context

This tutorial shows you how to:

- Evaluate important considerations, such as synchronization directions for remote tables, when designing a remote schema.
- Add unique primary keys to consolidated and remote databases.
- Create an ODBC data source that connects MobiLink to an Adaptive Server Enterprise database.
- Set up synchronization between a consolidated database and a remote database using the *Create Synchronization Model Wizard*.
- Customize synchronization settings using SQL Central.
- Deploy a consolidated database and remote database using the *Deploy Synchronization Model Wizard*.
- Synchronize the remote client with the consolidated database.

The purpose of this tutorial is to mobilize data for a chain of bookstores. Each bookstore in this scenario is a remote synchronization environment. Each bookstore has a local SQL Anywhere database that is synchronized with the Adaptive Server Enterprise database at headquarters. Each bookstore can have several computers that access and manipulate data from the remote database.

This tutorial assumes that the pubs2 sample schema is installed on an Adaptive Server Enterprise 15.7 server. The pubs2 sample schema is provided with Adaptive Server Enterprise 15.0 and it is an optional part of the install. For this tutorial, it is used as the consolidated database. Information about this sample can be found in the Adaptive Server Enterprise documentation.

This tutorial uses the default **sa** account. When Adaptive Server Enterprise is installed, the **sa** account has a null password. This tutorial assumes you have changed the null password to a valid password.

1. Designing the Schemas [page 167]
   The pubs2 sample schema is used as the consolidated database schema. It contains information about stores, titles, authors, publishers, and sales. The following table provides a description of each table in the Adaptive Server Enterprise database:
2. Lesson 1: Preparing the Consolidated Database [page 168]
   In this lesson, you increase the size of the consolidated database for MobiLink synchronization.
3. Lesson 2: Adding Unique Keys to the Consolidated Database [page 170]
   In this lesson, unique primary keys are added to the consolidated database.
4. Lesson 3: Connecting with MobiLink [page 172]
   In this lesson, you create an ODBC data source that connects MobiLink to the consolidated database.
5. Lesson 4: Creating a MobiLink Project and Synchronization Model [page 174]
   In this lesson you connect to the consolidated database by creating a new MobiLink project. A synchronization model is created automatically when you create a new project.
6. Lesson 5: Modify a Synchronization Model [page 175]
   In this lesson, you modify a synchronization model.
7. Lesson 6: Deploying the Synchronization Model [page 178]
   The *Deploy Synchronization Model Wizard* allows you to deploy the consolidated database and remote database. You can deploy each of these databases individually or you can deploy both of them. The *Deploy Synchronization Model Wizard* takes you through the steps of configuring options for deployment.
8. Lesson 7: Starting the Server and Client [page 179]
   In this lesson, you start the MobiLink server and remote database.
9. Lesson 8: Setting the Remote ID [page 181]
   In the remote schema, each remote database represents one store. The synchronization scripts include logic that instructs the MobiLink server to download a subset of data based on the remote ID of the remote database. You must set the database's remote ID to the value of a valid store identifier.
10. Lesson 9: Synchronizing [page 183]
    In this lesson, you synchronize the remote client for the first time using the dbmlsync utility.
11. Lesson 10: Viewing the Data in the Remote Database [page 185]
    After successfully synchronizing the remote client to the consolidated database through the MobiLink server, the remote data should be populated with information relevant to one store. You can verify the contents of the remote database in SQL Central using the SQL Anywhere 17 plug-in.
12. Lesson 11: Cleaning Up [page 186]
    Regenerate the pubs2 database and remove all tutorial materials from your computer.

Related Information

MobiLink Synchronization [page 4]

1.5.4.1 Designing the Schemas

The pubs2 sample schema is used as the consolidated database schema. It contains information about stores, titles, authors, publishers, and sales. The following table provides a description of each table in the Adaptive Server Enterprise database:

Table	Description
au_pix	Pictures of the authors.
authors	The authors of the various titles in the system.
discounts	Records of various discounts at particular stores.
sales	Each sale record is one sale made by a particular store.
salesdetail	Information about the different titles that were included in a particular sale.
stores	Each store record is one store or branch office in the system.
titleauthor	Information about which titles were written by which authors.
titles	Records of all the different books in the system.
blurbs, publishers, and roysched	Information that is not needed in this tutorial.

Designing the Remote Schema

It is unnecessary and inefficient for each store to have a copy of the entire consolidated database. The remote schema uses the same table names, but only contains information relevant to one particular store. To achieve this configuration, the remote schema is designed as a subset of the consolidated database in the following way:

Consolidated table	Remote table
au_pix	Includes all rows.
authors	Includes all rows.
discounts	Filter by stor_id.
sales	Filter by stor_id.
salesdetail	Filter by stor_id.
stores	Filter by stor_id.
titleauthor	Includes all rows.
titles	Includes all rows.
blurbs	Not included on remote.
publishers	Not included on remote.
roysched	Not included on remote.

Each store needs to keep records of all titles and authors so customers can search the store inventory. However, a bookstore does not need information about publishers or royalties, so this information is not synchronized to each store. Each store needs information about sales and discounts, but not about sales and discounts related to other stores. This behavior is achieved by filtering rows based on a store identifier.

> **i Note**
>
> You can also take a subset of columns from a table if certain columns are not required on the remote databases.

The next step is to choose the synchronization direction of each table. Consider what information a remote database needs to read and what information a remote database needs to create, change, or remove. In this example, a bookstore needs access to the list of authors and titles, but never enters a new author into the system. This places a restriction that authors and titles must always enter the system from the consolidated database at headquarters. However, a bookstore needs to record new sales on a regular basis. These factors lead to the following synchronization directions for the tables:

Table	Synchronization
titleauthor	Download to remote database only.
authors	Download to remote database only.
au_pix	Download to remote database only.
titles	Download to remote database only.
stores	Download to remote database only.
discounts	Download to remote database only.
sales	Download and upload.
salesdetail	Download and upload.

Parent topic: Tutorial: Using MobiLink with an Adaptive Server Enterprise Consolidated Database [page 165]

Next task: Lesson 1: Preparing the Consolidated Database [page 168]

1.5.4.2 Lesson 1: Preparing the Consolidated Database

In this lesson, you increase the size of the consolidated database for MobiLink synchronization.

Prerequisites

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

Context

MobiLink needs to add system tables and other objects to the pubs2 database for synchronization. When you add these objects, the size of the pubs2 database must be increased.

Procedure

1. Connect to the pubs2 database as **sa**, using isql in Adaptive Server Enterprise. At a command prompt, run the following command, all on one line:

   ```
   isql
   -U sa
   -P your-password-for-sa-account
   -D pubs2
   ```

   If you are accessing Adaptive Server Enterprise remotely, use the -S option to specify the server name.
2. To have proper permission for increasing the size of a database, you must access the master database. Run the following command in isql:

   ```
   use master
   go
   sp_dboption pubs2, "select into/bulkcopy/pllsort", true
   go
   ```
3. In Adaptive Server Enterprise, a database is stored on a disk or a portion of a disk. To increase the size of the pubs2 database, execute the following statement (you must specify the disk where pubs2 is stored):

   ```
   ALTER DATABASE pubs2 ON disk-name = 33
   ```

Results

The size of the consolidated database for MobiLink synchronization is increased.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Adaptive Server Enterprise Consolidated Database [page 165]

Previous: Designing the Schemas [page 167]

Next task: Lesson 2: Adding Unique Keys to the Consolidated Database [page 170]

Related Information

Adaptive Server Enterprise Consolidated Database

1.5.4.3 Lesson 2: Adding Unique Keys to the Consolidated Database

In this lesson, unique primary keys are added to the consolidated 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

In a synchronization system, the primary key of a table is the only way to uniquely identify a row in different databases and the only way to detect conflicts. Every table that is being mobilized must have a primary key. The primary key must never be updated. You must also guarantee that a primary key value inserted at one database is not inserted in another database.

There are several ways to generate unique primary keys. For simplicity, the method of composite primary keys is used in this tutorial. This method creates primary keys with multiple columns that are unique across the consolidated and remote databases.

Procedure

1. Connect to the pubs2 database as **sa**, using isql in Adaptive Server Enterprise. At a command prompt, run the following command, all on one line:

   ```
   isql
   -U sa
   -P your-password-for-sa-account
   -D pubs2
   ```

   If you are accessing Adaptive Server Enterprise remotely, use the -S option to specify the server name.
2. The following rows are not unique based on the composite primary key created for the salesdetail table. For simplicity, drop the rows by executing the following statements:

   ```
   DELETE FROM salesdetail
   WHERE stor_id = '5023'
   ```

```
AND ord_num = 'NF-123-ADS-642-9G3'
AND title_id = 'PC8888'
DELETE FROM salesdetail
WHERE stor_id = '5023'
AND ord_num = 'ZS-645-CAT-415-1B2'
AND title_id = 'BU2075'
```

3. The following indexes interfere with the creation of primary keys in a previous step. To drop the indexes, execute the following statements:

   ```
   DROP INDEX authors.auidind
   DROP INDEX titleauthor.taind
   DROP INDEX titles.titleidind
   DROP INDEX sales.salesind
   ```

4. Add unique primary keys by executing the following statements:

   ```
   ALTER TABLE au_pix ADD PRIMARY KEY (au_id)
   ALTER TABLE authors ADD PRIMARY KEY (au_id)
   ALTER TABLE titleauthor ADD PRIMARY KEY (au_id, title_id)
   ALTER TABLE titles ADD PRIMARY KEY (title_id)
   ALTER TABLE discounts ADD PRIMARY KEY (discounttype)
   ALTER TABLE stores ADD PRIMARY KEY (stor_id)
   ALTER TABLE sales ADD PRIMARY KEY (stor_id, ord_num)
   ALTER TABLE salesdetail ADD PRIMARY KEY (stor_id, ord_num, title_id)
   ```

   After executing these statements, the MobiLink server connects to the consolidated database and sets up synchronization for any number of remote databases.

   > **i Note**
   >
   > It is possible to synchronize data with consolidated databases that do not have primary keys. However, you must write you own synchronization events that act on shadow tables that are designed to identify rows uniquely in other tables.

   In a later lesson, the remote schema is created from the consolidated schema, so the remote schema has the same primary keys as the consolidated schema.

   Columns were specifically chosen to ensure unique primary keys for all databases. For the sales table, the primary key consists of the stor_id and ord_num columns. Any value inserted into the remote sales table must have a unique order number (the stor_id value is always the same). This practice ensures uniqueness in each remote sales table. The primary key in the consolidated sales table prevents conflicts if multiple stores upload data. Each upload from one store is unique to another store because their stor_id values are different.

   For the salesdetail table, the primary key consists of the stor_id, ord_num, and title_id columns. There may be multiple book titles in an order. For the remote sales tables, rows may have the same values for stor_id and ord_num, but they must have different title_id values. This configuration ensures uniqueness in each remote salesdetail table. Similar to the sales table, each upload to the consolidated database from a store is unique to another store because their stor_id values are different.

Results

Rows that are not unique are dropped and unique primary keys are added to the consolidated database.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Adaptive Server Enterprise Consolidated Database [page 165]

Previous task: Lesson 1: Preparing the Consolidated Database [page 168]

Next task: Lesson 3: Connecting with MobiLink [page 172]

Related Information

Unique Primary Keys

1.5.4.4 Lesson 3: Connecting with MobiLink

In this lesson, you create an ODBC data source that connects MobiLink to the consolidated 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. Create an ODBC data source.

   You should use the ODBC driver provided by Adaptive Server Enterprise. For this tutorial, use the following configuration settings:

General tab fields	Value
Data Source Name	ase_cons
Description	
Server Name (ASE Host Name)	localhost
Server Port	5000

General tab fields	Value
Database Name	pubs2
Logon ID	sa
Use Cursors	not selected

Transaction tab field	Value
Server Initiated Transactions	not selected

2. Test the ODBC connection:
   a. On the *General* tab, click *Test Connection*.

      The Adaptive Server Enterprise Logon screen appears.
   b. Enter the password for the `sa` account and click *OK*.

      The *Logon Succeeded* message appears.

Results

The ODBC data source is created and tested.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Adaptive Server Enterprise Consolidated Database [page 165]

Previous task: Lesson 2: Adding Unique Keys to the Consolidated Database [page 170]

Next task: Lesson 4: Creating a MobiLink Project and Synchronization Model [page 174]

Related Information

Recommended ODBC Drivers For MobiLink

1.5.4.5 Lesson 4: Creating a MobiLink Project and Synchronization Model

In this lesson you connect to the consolidated database by creating a new MobiLink project. A synchronization model is created automatically when you create a new project.

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. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Central*.
2. Click *Tools* > *MobiLink 17* > *New Project*.

   The *Create Project Wizard* appears.
3. In the *What do you want to name the new project* field, type **`ase_project`**. This is used as a folder name.
4. In the *Where do you want to save the new project* field, type `C:\mlase`. This is a folder name. Click *Next*.
5. In the *Database display name* field, type **`ase_cons`**.
6. Click *Edit*.
7. Perform the following tasks in the *Connect To A Generic ODBC Database* page:
   a. In the *User ID* field, type **`sa`**.
   b. In the *Password* field, type the password for the **sa** account.
   c. In the *ODBC Data Source name* field, click *Browse*, and choose **`ase_cons`**.
   d. Click *OK*, and click *Save*.
8. Select the *Remember the password* option if you specified a password, and click *Next*.
9. On the *New Remote Database Schema* page, in the *Which consolidated database tables and columns do you want to have in your remote database* list, choose the following tables:
   - au_pix
   - authors
   - discounts
   - sales
   - salesdetail
   - stores
   - titleauthor
   - titles

   Click *Next*.

10. If you receive a warning about fixed-length character columns, click *OK* and proceed with the next step.
11. Select the *Add a remote schema name to the project* option.
12. Type **`ase_remote_schema`** for the remote schema name. This is used as a folder name. Click *Next*.
13. Choose *SQL Anywhere* and then click *Finish*.

    If this is the first time the consolidated database has been used by MobiLink, a message appears asking you to install the MobiLink system setup. Installing the MobiLink system setup adds MobiLink system tables and procedures. Click *Yes*, and then click *OK*.

Results

The `ase_project` project and the `ase_remote_schema` synchronization model are created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Adaptive Server Enterprise Consolidated Database [page 165]

Previous task: Lesson 3: Connecting with MobiLink [page 172]

Next task: Lesson 5: Modify a Synchronization Model [page 175]

1.5.4.6 Lesson 5: Modify a Synchronization Model

In this lesson, you modify a synchronization model.

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. Right click the ase_remote_schema synchronization model and choose *Properties*.

2. Perform the following tasks:
    a. In the first field, type **`sync_ase`**.
    b. In the *Publication name* field, type **`sync_ase_publication`**.
    c. In the *Script version* field, type **`sync_ase_scriptversion`**.

       The publication is the object on the remote database that specifies what data is synchronized. MobiLink server scripts define how uploaded data from remotes should be applied to the consolidated database, and script versions group scripts. You can use different script versions for different applications, allowing you to maintain a single MobiLink server while synchronizing different applications.
    d. Click *Apply* and then click *OK*.
3. Set the direction that data is synchronized for each table in the synchronization model.

   Click the *Mappings* tab in the right pane, and set the rows in the *Mapping Direction* column as follows:
    - The **`sales`** and **`salesdetail`** tables should be set to *Bi-directional* (both upload and download).
    - The remaining tables should be set to *Download to remote only*.
4. Filter the rows downloaded to the remote database by remote ID.
    a. Select the row containing the **`stores`** table and then click the *Download Subset* tab.
    b. Change the *Subsetting strategy* to *Custom*.
    c. Filter the rows by remote ID, which uniquely identifies the remote database, by adding a restriction to the WHERE clause of the download_cursor script.

       Type a search condition in the *SQL expression to use in the download cursor's WHERE clause* field. For example, the following SQL script can be used for the **`stores`** table:

       ```
       "dbo"."stores"."stor_id" = {ml s.remote_id}
       ```

       The download cursor script specifies what columns and rows are downloaded from each table to the remote database. The search condition ensures that you only download information about one store, namely, the store that has an identifier that equals the remote ID for the database.
    d. Click the *Download Delete Subset* tab and change the *Download Delete Subset* from *Downloaded* to *All*.
5. Repeat the previous step for the rows containing the **`sales`**, **`salesdetail`**, and **`discounts`** tables.

   > i Note
   >
   > You must rename the table specified in the SQL script to the table name in the row that you are editing.
   >
   > Use the following WHERE clause script for the **`sales`** table:
   >
   > ```
   > "dbo"."sales"."stor_id" = {ml s.remote_id}
   > ```
   >
   > Use the following WHERE clause script for the **`salesdetail`** table:
   >
   > ```
   > "dbo"."salesdetail"."stor_id" = {ml s.remote_id}
   > ```
   >
   > Use the following WHERE clause script for the **`discounts`** table:
   >
   > ```
   > "dbo"."discounts"."stor_id" = {ml s.remote_id}
   > ```
6. Save the synchronization model.

Click *File* > *Save*.

Results

The synchronization model is complete and ready for deployment.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Adaptive Server Enterprise Consolidated Database [page 165]

Previous task: Lesson 4: Creating a MobiLink Project and Synchronization Model [page 174]

Next task: Lesson 6: Deploying the Synchronization Model [page 178]

Related Information

Consolidated Database Setup
MobiLink Server System Tables
MobiLink Server System Procedures
download_delete_cursor Scripts
Conflict Handling Overview
Conflict Resolution
Publications
Synchronization Model Tasks [page 41]
Table and Column Mappings [page 42]
Modifying the Download Strategy [page 47]
Modifying Conflict Detection and Resolution [page 55]

1.5.4.7 Lesson 6: Deploying the Synchronization Model

The *Deploy Synchronization Model Wizard* allows you to deploy the consolidated database and remote database. You can deploy each of these databases individually or you can deploy both of them. The *Deploy Synchronization Model Wizard* takes you through the steps of configuring options for deployment.

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 under *MobiLink 17*, expand **`ase_project`**, *Synchronization Models*, **`sync_ase`**.
2. Click *File* > *Deploy*.
3. Accept the default setting for the *Select the folder to contain files generated by the wizard* field and click *Next*.
4. Accept the default settings for the *Client Network Options* page and click *Next*.
5. On the *MobiLink User and Password* page under *What MobiLink user and password do you want to use*, select *Use these* and perform the following tasks:
   - In the *MobiLink user* field, type **`ase_remote`**.
   - In the *MobiLink password* field, type **`ase_pass`**.
   - Check *Register this user in the consolidated database. Registered users are permitted to synchronize* and click *Next*.
6. On the *Synchronization Profile* page, type **`sync_ase_profile`** in the *Synchronization profile name* field and click *Next*.
7. Perform the following tasks on the *Choose How To Prepare Databases For Synchronization* page:
   a. For *What do you want to do with the SQL script created to prepare the consolidated database for synchronization*, select *Execute against consolidated database*.
   b. For *What do you want to do with the SQL script created to prepare the remote database for synchronization*, select *Execute against a new remote database*.
   c. In the *User ID* field, type **`sa`**.
   d. In the *Password* and *Retype Password* fields, type **`passwd`** and click *Next*.
8. Click *Close* if the *Generating Deployment Files* window appears.
9. Click *Finish*.
10. Click *Close*.

Results

Your consolidated database is fully configured for synchronization with many remote clients, and you have successfully deployed one remote client. To deploy other remote clients, you can run this wizard again, making sure to create a new MobiLink user and opt out of deploying the consolidated database and MobiLink server. Since they have already been deployed, all you need to do is deploy other remote synchronization clients.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Adaptive Server Enterprise Consolidated Database [page 165]

Previous task: Lesson 5: Modify a Synchronization Model [page 175]

Next task: Lesson 7: Starting the Server and Client [page 179]

Related Information

Synchronization Model Deployment [page 58]
MobiLink Users in a Synchronization System
Using a SQL Anywhere Database as a Remote Database

1.5.4.8 Lesson 7: Starting the Server and Client

In this lesson, you start the MobiLink server and remote 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

In a previous lesson, you modified the download cursor script to download information related to one store. In this lesson, you specify the store by setting the remote ID to the store identifier.

Procedure

1. At a command prompt, navigate to the folder where you created the synchronization model. This is the root directory you chose in step 4 of the *Create Project Wizard* in lesson 4.

   If you used the suggested names from previous lessons, then navigate to the following directory: `C:\mlase\ase_project\sync_ase_deploy`.
2. To start the MobiLink server, run the following command:

   ```
   mlsrv.bat "DSN=ase_cons;UID=sa;PWD=passwd;"
   ```

   mlsrv.bat

   is the command file to start the MobiLink server.

   DSN

   is your ODBC data source name.

   UID

   is the user name you use to connect to the consolidated database.

   PWD

   is the password you use to connect as the specified user.

   If the MobiLink server fails to start, check the connection information for your consolidated database.
3. At a command prompt, navigate to the directory where the *Deploy Synchronization Model Wizard* created your remote database.

   If you used the suggested directory names, navigate to the following directory: `C:\mlase\ase_project\sync_ase_deploy`.
4. To start your remote SQL Anywhere database, run the following command:

   ```
   dbsrv17 -n remote_eng sync_ase_remote.db -n remote_db
   ```

   dbsrv17

   is the database server used to start the SQL Anywhere database.

   remote_eng

   is the database server name.

   sync_ase_remote.db

   is the database file that is started on remote_eng.

   remote_db

   is the name of the database on remote_eng.

Results

A SQL Anywhere database server named remote_eng starts and loads the database called remote_db.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Adaptive Server Enterprise Consolidated Database [page 165]

Previous task: Lesson 6: Deploying the Synchronization Model [page 178]

Next task: Lesson 8: Setting the Remote ID [page 181]

Related Information

Deployed Synchronization Models [page 66]
MobiLink Server
SQL Anywhere Database Server Executable (dbsrv17, dbeng17)

1.5.4.9 Lesson 8: Setting the Remote ID

In the remote schema, each remote database represents one store. The synchronization scripts include logic that instructs the MobiLink server to download a subset of data based on the remote ID of the remote database. You must set the database's remote ID to the value of a valid store identifier.

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

It is important to complete this step before the first synchronization because when the remote device synchronizes for the first time, it downloads all information related to the store (in this case, Thoreau Reading Discount Chain).

Procedure

1. Choose a valid store identifier.
    a. Connect to the pubs2 database as **sa**, using isql in Adaptive Server Enterprise. At a command prompt, run the following command, all on one line:

    ```
    isql
    -U sa
    -P your-password-for-sa-account
    -D pubs2
    ```

    If you are accessing Adaptive Server Enterprise remotely, use the -S option to specify the server name.
    b. To view a list of valid store identifiers in the stores table, execute the following statement:

    ```
    SELECT * FROM stores
    ```

    In this tutorial, the remote database represents the Thoreau Reading Discount Chain store, which has a value of 5023 for its store identifier.
    c. To exit isql, run the following command:

    ```
    exit
    ```
2. To set the database's remote ID to 5023, run the following command, all on one line:

    ```
    dbisql
        -c "SERVER=remote_eng;DBN=remote_db;UID=DBA;PWD=sql"
        "SET OPTION PUBLIC.ml_remote_id='5023'"
    ```

    dbisql

    is the application used to execute SQL commands against a SQL Anywhere database.

    server

    specifies the database server name remote_eng.

    dbn

    specifies the database name remote_db.

    uid

    specifies the user name used to connect to your remote database.

    pwd

    specifies the password used to connect to your remote database.

    SET OPTION PUBLIC.ml_remote_id='5023'

    is the SQL command used to set the remote ID to 5023.

Results

The database's remote ID to 5023, which is the value of the store identifier.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Adaptive Server Enterprise Consolidated Database [page 165]

Previous task: Lesson 7: Starting the Server and Client [page 179]

Next task: Lesson 9: Synchronizing [page 183]

Related Information

Remote IDs

1.5.4.10 Lesson 9: Synchronizing

In this lesson, you synchronize the remote client for the first time using the dbmlsync 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

Dbmlsync connects to the remote database, loads the synchronization information from the remote database, scans the transaction logs, and then generates the upload data. Then dbmlsync connects to the MobiLink server, authenticates itself with the MobiLink server, and then performs all the uploads and downloads necessary to synchronize the remote and consolidated databases based on a publication in the remote database.

Procedure

At a command prompt, run the following command, all on one line:

```
dbmlsync -c "SERVER=remote_eng;DBN=remote_db;UID=DBA;PWD=sql;"
```

```
    -n sync_ase_publication
    -u ase_remote -mp ase_pass
```

dbmlsync

is the synchronization application.

SERVER

specifies the name of the remote database server.

DBN

specifies the name of the remote database.

UID

specifies the user name used to connect to the remote database.

PWD

specifies the password used to connect to the remote database.

sync_ase_publication

is the name of the publication on the remote device that is used to perform the synchronization. (This publication was created using the *Create Synchronization Model Wizard*.)

ase_remote

is the user name used to authenticate with the MobiLink server.

ase_pass

is the password used to authenticate with the MobiLink server.

i Note

If you are running the dbmlsync application on a different computer from your MobiLink server, you must also pass in arguments that specify the location of the MobiLink server.

Results

The progress of the synchronization appears in the *SQL Anywhere MobiLink Client Messages* window. When this command runs successfully, the dbmlsync application populates the remote database with a subset of information from the consolidated database.

If synchronization fails, check the connection information you pass to the dbmlsync application, and the MobiLink user name and password. If the problem persists, check the publication name you used, and ensure the consolidated database and MobiLink server are running. You can also examine the contents of the synchronization logs (server and client).

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Adaptive Server Enterprise Consolidated Database [page 165]

Previous task: Lesson 8: Setting the Remote ID [page 181]

Next task: Lesson 10: Viewing the Data in the Remote Database [page 185]

Related Information

The Synchronization Process [page 18]
MobiLink SQL Anywhere Client Utility (dbmlsync) Syntax

1.5.4.11 Lesson 10: Viewing the Data in the Remote Database

After successfully synchronizing the remote client to the consolidated database through the MobiLink server, the remote data should be populated with information relevant to one store. You can verify the contents of the remote database in SQL Central using the SQL Anywhere 17 plug-in.

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 SQL Central.
2. Connect to the remote database:
    a. In the left pane, right-click *SQL Anywhere 17*, and choose *Connect*.
    b. In the *Authentication* dropdown list, choose *Database*, and perform the following steps:
        1. In the *User ID* field, type **`DBA`**.
        2. In the *Password* field, type **`passwd`**.
    c. From the *Action* dropdown list, choose *Connect to a running database on this computer*.
    d. In the *Server name* field, type **`remote_eng`** and in the *Database name* field, type **`remote_db`**.
    e. Click *Connect*.
3. If the tables created from the consolidated database are not visible, perform the following steps:
    a. Right-click `remote_db`, and *Configure Owner Filter*.
    b. Choose *dbo*, and click *OK*.

The tables created from the consolidated database appear in the left pane. Ownership of these tables by dbo is preserved in the remote database.

4. Choose any remote table, and click the *Data* tab in the right pane.

   In the sales, salesdetail, and stores tables, all the records are for the store with an identifier of 5023. This particular store is not concerned with the sales information of other stores. For this reason, you set the synchronization scripts to filter out rows by the remote ID, and you set this database's remote ID to the value of a particular store identifier. Now this particular store's database takes up less space, and requires less time to synchronize. Since the remote database size is kept to a minimum, frequently performed operations such as entering a new sale or processing a refund on a previous sale run faster and more efficiently.

Results

The data in the remote database is displayed.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using MobiLink with an Adaptive Server Enterprise Consolidated Database [page 165]

Previous task: Lesson 9: Synchronizing [page 183]

Next task: Lesson 11: Cleaning Up [page 186]

1.5.4.12 Lesson 11: Cleaning Up

Regenerate the pubs2 database and remove all tutorial materials from your computer.

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. Regenerate the pubs2 database.

   To run the script that installs the pubs2 database, run the following command:

   ```
   isql
   -U sa
   -P your-password-for-sa-account
   -i %SYBASE%\%SYBASE_ASE%\scripts\instpbs2
   ```

   If you are accessing Adaptive Server Enterprise remotely, use the -S option to specify the server name. You also have to copy the instpbs2 file locally onto your computer. The -i option needs to updated so that the new location of the instpbs2 file is specified.
2. Delete the synchronization model.
   1. Start SQL Central.
   2. Double-click *MobiLink 17* in the right pane.

      The **`sync_ase`** model appears.
   3. Right-click **`sync_ase`** and choose *Delete*.
3. Erase the remote database by using the dberase utility.

   Run the following command:

   ```
   dberase sync_ase\remote\sync_ase_remote.db
   ```

Results

The pubs2 database is regenerated and all tutorial materials are removed from your computer.

Task overview: Tutorial: Using MobiLink with an Adaptive Server Enterprise Consolidated Database [page 165]

Previous task: Lesson 10: Viewing the Data in the Remote Database [page 185]

1.5.5 Tutorial: Using Java or .NET for Custom User Authentication

MobiLink synchronization scripts can be written in SQL, Java, or .NET. You can use Java or .NET to add custom actions at any point of a synchronization.

Prerequisites

You require:

- Familiarity with Java or .NET
- Basic knowledge of MobiLink event scripts

The following software is required:

- SQL Anywhere 17
- Java Software Development Kit or the Microsoft .NET Framework

You must have the following privileges on the CustDB database:

- SYS_REPLICATION_ADMIN_ROLE system role
- CREATE ANY TRIGGER system privilege
- CREATE ANY VIEW system privilege
- EXECUTE ANY PROCEDURE system privilege

Context

In this tutorial, you add a Java or .NET method for the authenticate_user connection event. The authenticate_user event allows you to specify a custom authentication scheme and override the MobiLink built-in client authentication.

The goal for this tutorial is for you to gain competence and familiarity with MobiLink custom authentication. This tutorial shows you how to:

- Compile a source file with MobiLink server API references
- Specify class methods for particular table-level events
- Run the MobiLink server (mlsrv17) with the -sl option
- Test synchronization with a sample Windows client application

1. Lesson 1: Creating a Java Class for Custom Authentication (Server-Side) [page 189]
   In this lesson, you compile a class containing Java logic for custom authentication.
2. Lesson 2: Creating a .NET Class for Custom Authentication (Server-Side) [page 191]
   In this lesson, you compile a class containing .NET logic for custom authentication.
3. Lesson 3: Registering Your Java or .NETSscripts for the authenticate_user Event [page 193]
   SQL Anywhere includes a sample database (CustDB) that is set up for synchronization. The CustDB ULCustomer table, for example, is a synchronized table supporting a variety of table-level scripts. In

this lesson, you register the MobiLinkAuth authenticateUser method for the authenticate_user synchronization event. You add this script to CustDB.

4. Lesson 4: Starting the MobiLink Server [page 195]
   In this lesson, you run the MobiLink server with the -sl option to specify a set of directories to search for compiled files.
5. Lesson 5: Testing the Authentication [page 196]
   UltraLite comes with a sample Windows client that can initiate synchronization. In this lesson, you run the application against the CustDB consolidated database you started in the previous lesson.
6. Lesson 6: Cleaning Up [page 197]
   Remove all tutorial materials from your computer and reset the database for the Windows Sample Application.

Related Information

User Authentication Mechanisms
Synchronization Scripts in Microsoft .NET
Synchronization Script Writing in Java

1.5.5.1 Lesson 1: Creating a Java Class for Custom Authentication (Server-Side)

In this lesson, you compile a class containing Java logic for custom authentication.

Prerequisites

The MobiLink server must have access to the classes in `mlscript.jar` to execute Java synchronization logic. `mlscript.jar` contains a repository of MobiLink Java server API classes to utilize in your Java methods. When you compile your Java class, you reference `mlscript.jar`.

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

Context

To create a .NET class for customer authentication, see Creating a .NET class for custom authentication (server-side).

Procedure

1. Create a class named MobiLinkAuth and write an authenticateUser method.

   The MobiLinkAuth class includes the authenticateUser method used for the authenticate_user synchronization event. The authenticate_user event provides parameters for the user and password. You return the authentication result in the authentication_status inout parameter.

   > **i Note**
   >
   > You register the authenticateUser method for the authenticate_user synchronization event in Lesson 2: Registering your Java or .NET scripts for the authenticate_user event.

   Use the following code for your server application:

   ```
   import com.sap.ml.script.*;
   public class MobiLinkAuth {
       public void authenticateUser (
           com.sap.ml.script.InOutInteger authentication_status,
           String user,
           String pwd,
           String newPwd ) {
           if (user.startsWith("128")) {
               // success: an auth status code of 1000
               authentication_status.setValue(1000);
           } else {
               // fail: an authentication_status code of 4000
               authentication_status.setValue(4000);
           }
       }
   }
   ```

   This code illustrates a simple case of custom user authentication. Authentication succeeds when the client accesses the consolidated database using a user name that starts with **`128`**.
2. Save your code. This tutorial assumes `c:\MLauth` as the working directory for server-side components. Save the file as `MobiLinkAuth.java` in this directory.
3. Compile your class file.
   a. Navigate to the directory that contains your Java file.
   b. Compile the MobiLinkAuth class and refer to the MobiLink server Java API library.

      Run the following command, replacing *C:\Program Files\SQL Anywhere 17*\ with your SQL Anywhere 17 directory:

      ```
      javac MobiLinkAuth.java -classpath "C:\Program Files\SQL Anywhere 17\java
      \mlscript.jar"
      ```

Results

The `MobiLinkAuth.class` file is generated.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Java or .NET for Custom User Authentication [page 188]

Next task: Lesson 2: Creating a .NET Class for Custom Authentication (Server-Side) [page 191]

Related Information

Java and .NET User Authentication
authenticate_user Connection Event
Java Synchronization Example
.NET Synchronization Example

1.5.5.2 Lesson 2: Creating a .NET Class for Custom Authentication (Server-Side)

In this lesson, you compile a class containing .NET logic for custom authentication.

Prerequisites

The MobiLink server must have access to the classes in `Sap.MobiLink.Script.dll` to execute .NET synchronization logic. `Sap.MobiLink.Script.dll` contains a repository of MobiLink .NET server API classes to utilize in your .NET methods. When you compile your .NET class, you reference `Sap.MobiLink.Script.dll`.

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

Context

To create a .NET class for customer authentication, see Creating a Java class for custom authentication (server-side).

Procedure

1. Create a class named MobiLinkAuth and write an authenticateUser method.

   The MobiLinkAuth class includes the authenticateUser method used for the authenticate_user synchronization event. The authenticate_user event provides parameters for the user and password. You return the authentication result in the authentication_status inout parameter.

   > i Note
   >
   > You register the authenticateUser method for the authenticate_user synchronization event in Lesson 2: Registering your Java or .NET scripts for the authenticate_user event.

   Use the following code for your server application:

   ```
   using Sap.MobiLink.Script;
   public class MobiLinkAuth {
       public void authenticateUser(
           ref int authentication_status,
           string user,
           string pwd,
           string newPwd ) {
           if(user.StartsWith("128")) {
               // success: an auth status code of 1000
               authentication_status = 1000;
           } else {
               // fail: and authentication_status code of 4000
               authentication_status = 4000;
           }
       }
   }
   ```

   This code illustrates a simple case of custom user authentication. Authentication succeeds when the client accesses the consolidated database using a user name that starts with **128**.
2. Save your code. This tutorial assumes `c:\MLauth` as the working directory for server-side components. Save the file as `MobiLinkAuth.cs` in this directory.
3. Compile your class file.
   a. Navigate to the directory that contains your C# file.
   b. Compile the MobiLinkAuth class and refer to the MobiLink server .NET API library.

      Run the following command, replacing *C:\Program Files\SQL Anywhere 17*\ with your SQL Anywhere 17 directory:

      ```
      csc /out:MobiLinkAuth.dll /target:library /reference:"C:\Program Files\SQL
      Anywhere 17\Assembly\v3.5\Sap.MobiLink.Script.dll" MobiLinkAuth.cs
      ```

Results

The `MobiLinkAuth.dll` assembly is generated.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Java or .NET for Custom User Authentication [page 188]

Previous task: Lesson 1: Creating a Java Class for Custom Authentication (Server-Side) [page 189]

Next task: Lesson 3: Registering Your Java or .NETSscripts for the authenticate_user Event [page 193]

Related Information

Java and .NET User Authentication
authenticate_user Connection Event
Java Synchronization Example
.NET Synchronization Example

1.5.5.3 Lesson 3: Registering Your Java or .NETSscripts for the authenticate_user Event

SQL Anywhere includes a sample database (CustDB) that is set up for synchronization. The CustDB ULCustomer table, for example, is a synchronized table supporting a variety of table-level scripts. In this lesson, you register the MobiLinkAuth authenticateUser method for the authenticate_user synchronization event. You add this script to CustDB.

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

CustDB is designed to be a consolidated database server for both UltraLite and SQL Anywhere clients. The CustDB database has an ODBC data source named SQL Anywhere 17 CustDB.

Procedure

1. Connect to the sample database from Interactive SQL.

   Run the following command:

   ```
   dbisql -c "DSN=SQL Anywhere 17 CustDB;uid=ml_server;pwd=sql"
   ```

2. Use the ml_add_java_connection_script or ml_add_dnet_connection_script stored procedure to register the authenticateUser method for the authenticate_user event.

   For Java, execute the following SQL statements:

   ```
   CALL ml_server.ml_add_java_connection_script(
       'custdb 17.0',
       'authenticate_user',
       'MobiLinkAuth.authenticateUser');
   COMMIT;
   ```

   For .NET, execute the following SQL statements:

   ```
   CALL ml_add_dnet_connection_script(
       'custdb 17.0',
       'authenticate_user',
       'MobiLinkAuth.authenticateUser');
   COMMIT;
   ```

Results

The authenticateUser method is registered for the authenticate_user event.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Java or .NET for Custom User Authentication [page 188]

Previous task: Lesson 2: Creating a .NET Class for Custom Authentication (Server-Side) [page 191]

Next task: Lesson 4: Starting the MobiLink Server [page 195]

Related Information

Script Additions and Deletions
Synchronization Script Writing in Java

Synchronization Scripts in Microsoft .NET
Java Class Debugging
Synchronization Scripts
Synchronization Events
Setting Break Points to Debug .NET Synchronization Logic
Java Synchronization Example
.NET Synchronization Example
ml_add_java_connection_script System Procedure
ml_add_dnet_connection_script System Procedure

1.5.5.4 Lesson 4: Starting the MobiLink Server

In this lesson, you run the MobiLink server with the -sl option to specify a set of directories to search for compiled files.

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

Connect to the CustDB sample database and load your Java class or .NET assembly on the mlsrv17 command line.

Replace `c:\MLauth` with the location of your source files.

For Java, run the following command:

```
mlsrv17 -c "DSN=SQL Anywhere 17 CustDB;uid=ml_server;pwd=sql" -o serverOut.txt -v
+ -sl java(-cp c:\MLauth)
```

For .NET, run the following command:

```
mlsrv17 -c "DSN=SQL Anywhere 17 CustDB;uid=ml_server;pwd=sql" -o serverOut.txt -v
+ -sl dnet(-MLAutoLoadPath=c:\MLauth)
```

Results

The MobiLinkAuth method is executed when the authenticate_user synchronization event occurs.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Java or .NET for Custom User Authentication [page 188]

Previous task: Lesson 3: Registering Your Java or .NETSscripts for the authenticate_user Event [page 193]

Next task: Lesson 5: Testing the Authentication [page 196]

Related Information

MobiLink Server Options
-sl java mlsrv17 Option
-sl dnet mlsrv17 Option

1.5.5.5 Lesson 5: Testing the Authentication

UltraLite comes with a sample Windows client that can initiate synchronization. In this lesson, you run the application against the CustDB consolidated database you started in the previous lesson.

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 sample application.

   Click *Start* > *Programs* > *SQL Anywhere 17* > *UltraLite* > *Windows Sample Application*.
2. Enter an invalid employee ID and synchronize.

   In this application, the employee ID is also the MobiLink user name. If the user name does not begin with `128`, your logic causes synchronization to fail. Enter a value of `50` for the employee ID and click *OK*.

Results

An error stating that the authenticate_user script returned 4000 appears in the MobiLink server messages window.

A SQLCODE -1497 synchronization error indicating an invalid user ID or password appears in the *UltraLite CustDB Demo* window.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Java or .NET for Custom User Authentication [page 188]

Previous task: Lesson 4: Starting the MobiLink Server [page 195]

Next task: Lesson 6: Cleaning Up [page 197]

Related Information

CustDB Sample for MobiLink [page 71]
Invalid user ID or password

1.5.5.6 Lesson 6: Cleaning Up

Remove all tutorial materials from your computer and reset the database for the Windows Sample Application.

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. Delete your Java or .NET source files.

For example, delete the `c:\MLauth` directory.

> ⚠ Caution
>
> Make sure you only have tutorial related materials in this directory.

2. Close Interactive SQL and the UltraLite Windows client application.

   Click *File* > *Exit* in each application.
3. Close the SQL Anywhere, MobiLink, and synchronization client windows.

   Right-click each task bar item and then click *Close*.
4. Reset the database for the Windows Sample Application.

   Run the following command from the *%SQLANYSAMP17%*`\UltraLite\CustDB` directory:

   ```
   makedbs
   ```

Results

All tutorial materials are removed from your computer and the database for the Windows Sample Application is reset.

Task overview: Tutorial: Using Java or .NET for Custom User Authentication [page 188]

Previous task: Lesson 5: Testing the Authentication [page 196]

1.5.6 Tutorial: Using Direct Row Handling

You can use direct row handling to communicate remote data to any central data source, application, or web service other than a supported consolidated database.

Prerequisites

You require:

- Familiarity with Java or .NET
- Basic knowledge of MobiLink event scripts

The following software is required:

- SQL Anywhere 17
- Java Software Development Kit or the Microsoft .NET Framework

You must have the following roles and privileges on the consolidated database:

- SYS_AUTH_RESOURCE_ROLE compatibility role
- MONITOR system privilege

You must have the following roles and privileges on the remote database:

- SYS_REPLICATION_ADMIN_ROLE system role
- SYS_RUN_REPLICATION_ROLE system role

Context

In this tutorial you learn how to use the MobiLink server APIs for Java or .NET for simple direct row handling. You also learn how to synchronize the client RemoteOrders table with the consolidated database and add special direct row handling processing for the OrderComments table.

This tutorial shows you how to:

- Use the MobiLink server APIs for Java or .NET
- Create methods for MobiLink direct row handling

1. Lesson 1: Setting up a Text File Data Source [page 200]
   In this lesson, you create a new text file to store order information.
2. Lesson 2: Setting up Your MobiLink Consolidated Database [page 201]
   In this lesson, you create a database and define an ODBC data source.
3. Lesson 3: Creating a Table in Your MobiLink Consolidated Database [page 203]
   In this lesson, you create the RemoteOrders table in the MobiLink consolidated database.
4. Lesson 4: Adding Synchronization Scripts [page 204]
   You use direct row handling to add synchronization script information to your MobiLink consolidated database using stored procedures. In this lesson, you register method names corresponding to the handle_UploadData, handle_DownloadData, end_download, download_cursor, and download_delete_cursor events. You create your own Java or .NET class in a later lesson.
5. Lesson 5: Creating a Java or .NET Class for MobiLink Direct Row Handling [page 208]
   In this lesson, you use direct row handling to process rows in the OrderComments table in your client database.
6. Lesson 6: Starting the MobiLink Server [page 218]
   In this lesson, you start the MobiLink server. Start the MobiLink server (mlsrv17) using the -c option to connect to your consolidated database. Use the -sl java or -sl dnet option to load your Java or .NET class, respectively.
7. Lesson 7: Setting up Your MobiLink Client Database [page 220]
   In this lesson, you use a SQL Anywhere database for your consolidated database and your MobiLink client. For tutorial purposes, your MobiLink client, consolidated database, and MobiLink server all reside on the same computer.
8. Lesson 8: Synchronizing [page 222]
   The dbmlsync utility initiates MobiLink synchronization for SQL Anywhere remote databases. Before starting dbmlsync, add order data and comments to your remote database.
9. Lesson 9: Cleaning Up [page 224]
   Remove the tutorial materials from your computer.

Related Information

MobiLink Synchronization [page 4]
Synchronization Techniques
Direct Row Handling
SAP SQL Anywhere Forum

1.5.6.1 Lesson 1: Setting up a Text File Data Source

In this lesson, you create a new text file to store order information.

Prerequisites

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

Procedure

1. Create a new blank text file.
2. Add the following tab-delimited values representing the comment_id, order_id, and order_comment to the file. At the end of each line, press Enter.

```
786    34    OK, ship promotional material.
787    35    Yes, the product is going out of production.
788    36    No, your commission cannot be increased...
```

3. Save the file in your working directory. This tutorial assumes `c:\MLdirect` as the working directory for server-side components. Save the file as `orderResponses.txt` in this directory.

Results

The text file is created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Direct Row Handling [page 198]

Next task: Lesson 2: Setting up Your MobiLink Consolidated Database [page 201]

1.5.6.2 Lesson 2: Setting up Your MobiLink Consolidated Database

In this lesson, you create a database and define an ODBC data source.

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

Your MobiLink consolidated database is a central repository of data and includes MobiLink system tables and stored procedures used to manage the synchronization process. With direct row handling, you synchronize with a data source other than a consolidated database, but you still need a consolidated database to maintain information used by the MobiLink server.

> **i Note**
>
> If you already have a MobiLink consolidated database set up with MobiLink system objects and an ODBC data source, you can skip this lesson.

Procedure

1. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Central*.
2. Click *Tools* > *SQL Anywhere 17* > *Create Database*.
3. Click *Next*.
4. Accept the default value *Create a database on this computer* and click *Next*.
5. In the *Save the main database file to the following file* field, type the file name and path for the database. For example, `c:\MLdirect\MLconsolidated.db`.
6. Follow the remaining instructions in the *Create Database Wizard* and accept the default values. If prompted to specify a user ID and password for the DBA user, enter **`DBA`** and **`passwd`**, respectively..

   On the *Connect To The Database* page, clear the *Stop the database after last disconnect* option.

7. Click *Finish*.

   The MLconsolidated database appears in SQL Central.
8. Click *Close* on the *Creating Database* window if the window did not close automatically.
9. Use the SQL Anywhere 17 driver to define an ODBC data source for the MLconsolidated database.

   In SQL Central, click *Tools* > *SQL Anywhere 17* > *Open ODBC Administrator*.
10. Click the *User DSN* tab and click *Add*.
11. In the *Create New Data Source* window, click *SQL Anywhere 17* and click *Finish*.
12. Perform the following tasks in the *ODBC Configuration For SQL Anywhere* window:
    a. Click the *ODBC* tab.
    b. In the *Data source name* field, type **`mldirect_db`**.
    c. Click the *Login* tab.
    d. In the *User ID* field, type **`DBA`**.
    e. In the *Password* field, type **`passwd`**.
    f. In the *Server name* field, type **`MLconsolidated`**.
    g. Click *OK*.
13. Close the ODBC Data Source Administrator.

    Click *OK* on the *ODBC Data Source Administrator* window.

Results

The consolidated database and ODBC data source are created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Direct Row Handling [page 198]

Previous task: Lesson 1: Setting up a Text File Data Source [page 200]

Next task: Lesson 3: Creating a Table in Your MobiLink Consolidated Database [page 203]

Related Information

MobiLink Consolidated Databases
Initialization Utility (dbinit)

1.5.6.3 Lesson 3: Creating a Table in Your MobiLink Consolidated Database

In this lesson, you create the RemoteOrders table in the MobiLink consolidated 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

The RemoteOrders table you create contains the following columns:

order_id

A unique identifier for orders.

product_id

A unique identifier for products.

quantity

The number of items sold.

order_status

The order status.

last_modified

The last modification date of a row. You use this column for timestamp-based downloads, a common technique used to filter rows for efficient synchronization.

Procedure

1. Connect to your database from Interactive SQL.

   You can start Interactive SQL from SQL Central or at a command prompt.

   - To start Interactive SQL from SQL Central, right-click the **`MLconsolidated - DBA`** database and click *Open Interactive SQL*.
   - To start Interactive SQL at a command prompt, run the following command:

     ```
     dbisql -c "DSN=mldirect_db"
     ```

2. Execute the following SQL statement in Interactive SQL to create the RemoteOrders table.

   ```
   CREATE TABLE RemoteOrders (
   ```

```
    order_id            INTEGER NOT NULL,
    product_id          INTEGER NOT NULL,
    quantity            INTEGER,
    order_status        VARCHAR(10) DEFAULT 'new',
    last_modified       TIMESTAMP DEFAULT CURRENT TIMESTAMP,
    PRIMARY KEY(order_id)
);
```

Interactive SQL creates the RemoteOrders table in your consolidated database.

3. Execute the following statement in Interactive SQL to create MobiLink system tables and stored procedures.

   Replace *C:\Program Files\SQL Anywhere 17* with the location of your SQL Anywhere 17 installation.

   ```
   READ "C:\Program Files\SQL Anywhere 17\MobiLink\setup\syncsa.sql";
   ```

   Interactive SQL applies `syncsa.sql` to your consolidated database. Running `syncsa.sql` creates a series of system tables and stored procedures prefaced with **ml_**. The MobiLink server works with these tables and stored procedures in the synchronization process.

Results

The RemoteOrders table is created and MobiLink system tables and stored procedures are added to the consolidated database.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Direct Row Handling [page 198]

Previous task: Lesson 2: Setting up Your MobiLink Consolidated Database [page 201]

Next task: Lesson 4: Adding Synchronization Scripts [page 204]

Related Information

CREATE TABLE Statement

1.5.6.4 Lesson 4: Adding Synchronization Scripts

You use direct row handling to add synchronization script information to your MobiLink consolidated database using stored procedures. In this lesson, you register method names corresponding to the handle_UploadData, handle_DownloadData, end_download, download_cursor, and download_delete_cursor events. You create your own Java or .NET class in a later lesson.

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 row handling allows you to synchronize remote data with tables in your MobiLink consolidated database. SQL-based scripts define:

- How data that is uploaded from a MobiLink client is to be applied to the consolidated database.
- What data should be downloaded from the consolidated database.

The following SQL-based upload and download events are created:

upload_insert

This event defines how new orders inserted in a client database should be applied to the consolidated database.

download_cursor

This event defines the orders that should be downloaded to remote clients.

download_delete_cursor

This event is required when using synchronization scripts that are not upload-only. Set the MobiLink server to ignore this event for the purpose of this tutorial.

Procedure

1. Connect to your consolidated database from Interactive SQL, if you are not already connected, by running the following command:

   ```
   dbisql -c "DSN=mldirect_db"
   ```

2. Use the ml_add_table_script stored procedure to add SQL-based table scripts for the upload_insert, download_cursor and download_delete_cursor events.

   Execute the following SQL statements in Interactive SQL. The upload_insert script inserts the uploaded order_id, product_id, quantity, and order_status into the MobiLink consolidated database. The download_cursor script uses timestamp-based filtering to download updated rows to remote clients.

   ```
   CALL ml_add_table_script( 'default', 'RemoteOrders',
       'upload_insert',
       'INSERT INTO RemoteOrders( order_id, product_id, quantity, order_status)
   ```

```
     VALUES( {ml r.order_id}, {ml r.product_id}, {ml r.quantity}, {ml
r.order_status} )' );

CALL ml_add_table_script( 'default', 'RemoteOrders',
    'download_cursor',
    'SELECT order_id, product_id, quantity, order_status
     FROM RemoteOrders WHERE last_modified >= {ml s.last_table_download}');
CALL ml_add_table_script( 'default', 'RemoteOrders',
    'download_delete_cursor', '--{ml_ignore}');
COMMIT;
```

3. Register a Java or .NET method for the end_download event.

   You use this method to free memory resources when the MobiLink server runs the end_download connection event.

   For Java, execute the following statement in Interactive SQL:

```
CALL ml_add_java_connection_script( 'default',
    'end_synchronization',
    'MobiLinkOrders.EndSync' );
```

   For .NET, execute the following statement in Interactive SQL:

```
CALL ml_add_dnet_connection_script( 'default',
    'end_synchronization',
    'MobiLinkOrders.EndSync' );
```

   Interactive SQL registers the user-defined EndDownload method for the end_download event.
4. Register Java or .NET methods for the handle_UploadData and handle_DownloadData events.

   For Java, execute the following statements in Interactive SQL:

```
CALL ml_add_java_connection_script( 'default',
    'handle_UploadData',
    'MobiLinkOrders.GetUpload' );

CALL ml_add_java_connection_script( 'default',
    'handle_DownloadData',
    'MobiLinkOrders.SetDownload' );
```

   For .NET, execute the following statements in Interactive SQL:

```
CALL ml_add_dnet_connection_script( 'default',
    'handle_UploadData',
    'MobiLinkOrders.GetUpload' );

CALL ml_add_dnet_connection_script( 'default',
    'handle_DownloadData',
    'MobiLinkOrders.SetDownload' );
```

   Interactive SQL registers the user-defined GetUpload and SetDownload methods for the handle_UploadData and handle_DownloadData events, respectively. You create these methods in an upcoming lesson.
5. Register download_cursor and download_delete_cursor events.

   Execute the following statements in Interactive SQL:

```
CALL ml_add_table_script( 'default', 'OrderComments',
    'download_cursor', '--{ml_ignore}');
CALL ml_add_table_script( 'default', 'OrderComments',
    'download_delete_cursor', '--{ml_ignore}');
```

The download_cursor and download_delete_cursor events must be registered for the OrderComments table when using scripts because the synchronization is bi-directional and not upload-only.

6. Commit your changes.

   Execute the following statement in Interactive SQL:

   ```
   COMMIT;
   ```
7. Close Interactive SQL.

Results

Method names corresponding to the handle_UploadData, handle_DownloadData, end_download, download_cursor, and download_delete_cursor events are registered.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Direct Row Handling [page 198]

Previous task: Lesson 3: Creating a Table in Your MobiLink Consolidated Database [page 203]

Next task: Lesson 5: Creating a Java or .NET Class for MobiLink Direct Row Handling [page 208]

Related Information

Overview of MobiLink Events
Script Additions and Deletions
Scripts Required for Synchronization
Scripts to Upload Rows
Scripts to Download Rows
Direct Row Handling
Direct Uploads
Direct Downloads
Partitioned Rows Among Remote Databases
Implementing Timestamp-based Downloads
upload_insert Table Event
upload_update Table Event
upload_delete Table Event
download_cursor Table Event

download_delete_cursor Table Event

1.5.6.5 Lesson 5: Creating a Java or .NET Class for MobiLink Direct Row Handling

In this lesson, you use direct row handling to process rows in the OrderComments table in your client 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

You add the following methods for direct row handling:

GetUpload

Use this method for the handle_UploadData connection event. GetUpload writes uploaded comments to a file called `orderComments.txt`.

SetDownload

Use this method for the handle_DownloadData connection event. SetDownload uses the `orderResponses.txt` file to download responses to remote clients.

EndSync

Use this method for the end_synchronization connection event. EndSync frees memory resources.

The following procedure shows you how to create a Java or .NET class that includes your methods for processing.

Procedure

1. Create a class named MobiLinkOrders in Java or .NET.

   For Java, use the following code:

   ```
   import com.sap.ml.script.*;
   import java.io.*;
   import java.sql.*;

   public class MobiLinkOrders
   {
   ```

For .NET, use the following code:

```
using Sap.MobiLink.Script;
using System.IO;
using System.Data;
using System.Text;
public class MobiLinkOrders
{
```

2. Declare a class-level DBConnectionContext instance.

   For Java, use the following code:

```
DBConnectionContext _cc;
```

   For .NET, use the following code:

```
private DBConnectionContext _cc = null;
```

   The MobiLink server passes a DBConnectionContext instance to your class constructor. DBConnectionContext encapsulates information about the current connection with the MobiLink consolidated database.

3. Declare objects used for file input and output.

   For Java, declare a java.io.FileWriter and java.io.BufferedReader as follows:

```
FileWriter _myWriter;
BufferedReader _myReader;
```

   For .NET, declare a StreamWriter and StreamReader as follows:

```
private static StreamWriter _myWriter = null;
private static StreamReader _myReader = null;
```

4. Create your class constructor.

   Your class constructor sets your class-level DBConnectionContext instance.

   For Java, use the following code:

```
public MobiLinkOrders( DBConnectionContext cc )
    throws IOException, FileNotFoundException
{
    _cc = cc;
}
```

   For .NET, use the following code:

```
public MobiLinkOrders(DBConnectionContext cc)
{
    _cc = cc;
}
```

5. Write the GetUpload method for the handle_UploadData connection event.

   The GetUpload method obtains an UploadedTableData class instance representing the OrderComments table. The OrderComments table contains special comments made by remote sales employees. You create this table in a later lesson.

The UploadedTableData getInserts method returns a result set for new order comments. The writeOrderComment method writes out each row in the result set to a text file.

For Java, use the following code:

```
public void writeOrderComment( int commentID, int orderID, String comments )
    throws IOException
{
    if (_myWriter == null) {
        _myWriter = new FileWriter( "C:\\MLdirect\\orderComments.txt",true);
    }
    _myWriter.write(commentID + "\t" + orderID + "\t" + comments + "\n");
    _myWriter.flush();
}
public void GetUpload( UploadData ut )
    throws SQLException, IOException
{
    // Get an UploadedTableData for OrderComments
    UploadedTableData orderCommentsTbl =
ut.getUploadedTableByName("OrderComments");
    // Get inserts uploaded by the MobiLink client
    ResultSet insertResultSet = orderCommentsTbl.getInserts();
    while ( insertResultSet.next() )
    {
        // Get order comments
        int commentID = insertResultSet.getInt("comment_id");
        int orderID = insertResultSet.getInt("order_id");
        String orderComment = insertResultSet.getString("order_comment");
        if (orderComment != null) {
            writeOrderComment(commentID,orderID,orderComment);
        }
    }
    insertResultSet.close();
}
```

For .NET, use the following code:

```
public void WriteOrderComment(int commentID, int orderID, string comments)
{
    if (_myWriter == null) {
        _myWriter = new StreamWriter("c:\\MLdirect\\orderComments.txt");
    }
    _myWriter.WriteLine("{0}\t{1}\t{2}", commentID, orderID, comments);
    _myWriter.Flush();
}
public void GetUpload(UploadData ut)
{
    // Get UploadedTableData for remote table called OrderComments
    UploadedTableData order_comments_table_data =
        ut.GetUploadedTableByName("OrderComments");
    // Get inserts uploaded by the MobiLink client
    IDataReader new_comment_reader = order_comments_table_data.GetInserts();
    while (new_comment_reader.Read())
    {
        // Columns are
        // 0 - "comment_id"
        // 1 - "order_id"
        // 2 - "order_comment"
        // You can look up these values using the DataTable returned by:
        // order_comments_table_data.GetSchemaTable().
        // In this example, you just use the known column order to
        // determine the column indexes; alternatively, you could use
        // the column names

        // Only process this insert if the order_comment column is not null
        if (!new_comment_reader.IsDBNull(2)) {
            int commentID = new_comment_reader.GetInt32(0);
            int orderID = new_comment_reader.GetInt32(1);
            string orderComment = new_comment_reader.GetString(2);
            WriteOrderComment(commentID, orderID, orderComment);
        }
    }
    new_comment_reader.Close();
}
```

6. Write the SetDownload method:
    a. Obtain a class instance representing the OrderComments table.

        Use the DBConnectionContext getDownloadData method to obtain a DownloadData instance. Use the DownloadData getDownloadTableByName method to return a DownloadTableData instance for the OrderComments table.

        For Java, use the following code:

        ```
        public void SetDownload()
            throws SQLException, IOException
        {
            DownloadData download_d = _cc.getDownloadData();
            DownloadTableData download_td =
        download_d.getDownloadTableByName( "OrderComments" );
        ```

        For .NET, use the following code:

        ```
        private const string read_file_path = "c:\\MLdirect\\orderResponses.txt";
        public void SetDownload()
        {
            if ((_myReader == null) && !File.Exists(read_file_path)) {
                System.Console.Out.Write("There is no file to read.");
                return;
            }
            DownloadTableData comments_for_download =
                _cc.GetDownloadData().GetDownloadTableByName("OrderComments");
        ```

        > **i Note**
        >
        > You create the OrderComments table on the remote database in Lesson 7: Setting up your MobiLink client database.

    b. Obtain a prepared statement or IDbCommand that allows you to add, insert, or update operations to the download.

        For Java, use the DownloadTableData getUpsertPreparedStatement method to return a java.sql.PreparedStatement instance as follows:

        ```
        PreparedStatement update_ps = download_td.getUpsertPreparedStatement();
        ```

        For .NET, use the DownloadTableData GetUpsertCommand method as follows:

        ```
        // Add upserts to the set of operation that are going to be
        // applied at the remote database
        IDbCommand comments_upsert = comments_for_download.GetUpsertCommand();
        ```

    c. Set the download data for each row.

        This code traverses through the `orderResponses.txt` and adds data to the MobiLink download.

For Java, use the following code:

```
try {
    if (_myReader == null)
        _myReader = new BufferedReader(new FileReader(
            "C:\\MLdirect\\orderResponses.txt"));

    String commentLine = _myReader.readLine();

    while (commentLine != null) {
        // Get the next line from orderResponses.txt
        String[] responseDetails = commentLine.split("\t");
        if (responseDetails.length != 3) {
            System.err.println("Error reading from orderResponses.txt");
            System.err.println("Error setting direct row handling
download");
            return;
        }
        int commentID = Integer.parseInt(responseDetails[0]);
        int orderID = Integer.parseInt(responseDetails[1]);
        String updatedComment = responseDetails[2];
        // Set an order comment response in the MobiLink download
        update_ps.setInt(1, comment_id);
        update_ps.setInt(2, order_id);
        update_ps.setString(3, updated_comment);
        // Send comment response down to clients
        update_ps.executeUpdate();
        commentLine = _myReader.readLine();
    }
}
```

For .NET, use the following code:

```
if (_myReader == null) {
    _myReader = new StreamReader(read_file_path);
}
string commentLine;
while ((commentLine = _myReader.ReadLine()) != null) {
    // Three values are on each line separated by '\t'
    string[] responseDetails = comment_line.Split('\t');
    if (responseDetails.Length != 3) {
        throw (new SynchronizationException(
            "Error reading from orderResponses.txt"));
    }
    int commentID = System.Int32.Parse(responseDetails[0]);
    int orderID = System.Int32.Parse(responseDetails[1]);
    string updatedComment = responseDetails[2];
    // Parameters of the correct number and type have
    // already been added so you just need to set the
    // values of the IDataParameter
    ((IDataParameter)(comments_upsert.Parameters[0])).Value =
        commentID;
    ((IDataParameter)(comments_upsert.Parameters[1])).Value =
        orderID;
    ((IDataParameter)(comments_upsert.Parameters[2])).Value =
        updatedComment;
    // Add the upsert operation
    comments_upsert.ExecuteNonQuery();
}
```

d. Close the prepared statement used for adding insert or update operations to the download.

For Java, use the following code:

```
finally {
    update_ps.close();
}
```

For .NET, you do not need to close the IDbCommand. The object is destroyed automatically at the end of the download.

7. Write the EndSync method.

   This method handles the end_synchronization connection event and gives you an opportunity to free resources.

   For Java, use the following code:

   ```
   public void EndSync()
       throws IOException
   {
       if (_myReader != null) {
           _myReader.close();
           _myReader = null;
       }
       if (_myWriter != null) {
           _myWriter.close();
           _myWriter = null;
       }
   }
   ```

   For .NET, use the following code:

   ```
   public void EndSync()
   {
       if (_myWriter != null) {
           _myWriter.Close();
           _myWriter = null;
       }
       if (_myReader != null) {
           _myReader.Close();
           _myReader = null;
       }
   }
   ```

8. Save your code.

   For Java, save your code as `MobiLinkOrders.java` in your working directory. `c:\MLdirect`.

   For .NET, save your code as `MobiLinkOrders.cs` in your working directory. `c:\MLdirect`.

9. To verify the code, see Complete MobiLinkOrders code listing (Java) or Complete MobiLinkOrders code listing (.NET).

10. Compile your class file.

    a. Navigate to the directory containing your Java or .NET source files.
    b. Compile MobiLinkOrders and refer to the MobiLink server API library for Java or .NET.

       For Java, you need to reference `mlscript.jar`, located in *%SQLANY17%*`\java`. Run the following command:

       ```
       javac -classpath "%SQLANY17%\java\mlscript.jar" MobiLinkOrders.java
       ```

For .NET, you need to include `Sap.MobiLink.Script.dll`, located in *%SQLANY17%*`\Assembly\V3.5`. Run the following command:

```
csc /out:MobiLinkServerCode.dll /target:library /reference:"%SQLANY17%
\Assembly\V3.5\Sap.MobiLink.Script.dll" MobiLinkOrders.cs
```

> i Note
>
> This example does not ensure that primary key values are unique.

Results

The Java or .NET class for MobiLink direct row handling is created.

Next Steps

Proceed to the next lesson.

In this section:

Complete MobiLinkOrders Code Listing (Java) [page 215]
    The following is the complete MobiLinkOrders listing for Java direct row handling.

Complete MobiLinkOrders Code Listing (.NET) [page 216]
    The following is the complete MobiLinkOrders listing for .NET direct row handling.

Task overview: Tutorial: Using Direct Row Handling [page 198]

Previous task: Lesson 4: Adding Synchronization Scripts [page 204]

Next task: Lesson 6: Starting the MobiLink Server [page 218]

Related Information

Direct Row Handling
Synchronization Script Writing in Java
Synchronization Scripts in Microsoft .NET
Unique Primary Keys

1.5.6.5.1 Complete MobiLinkOrders Code Listing (Java)

The following is the complete MobiLinkOrders listing for Java direct row handling.

```
import com.sap.ml.script.*;
import java.io.*;
import java.sql.*;
public class MobiLinkOrders
{
    // Class level DBConnectionContext
    DBConnectionContext _cc;

    // Java objects for file i/o
    FileWriter _myWriter;
    BufferedReader _myReader;
    public MobiLinkOrders( DBConnectionContext cc )
        throws IOException, FileNotFoundException
    {
        _cc = cc;
    }

    public void writeOrderComment( int commentID, int orderID, String comments )
        throws IOException
    {
        if (_myWriter == null)
        {
            _myWriter = new FileWriter( "C:\\MLdirect\\orderResponses.txt",true);
        }
        _myWriter.write(commentID + "\t" + orderID + "\t" + comments + "\n");
        _myWriter.flush();
    }
    public void GetUpload( UploadData ut )
        throws SQLException, IOException
    {
        // Get an UploadedTableData for OrderComments
        UploadedTableData orderCommentsTbl =
ut.getUploadedTableByName("OrderComments");

        // Get inserts uploaded by the MobiLink client
        ResultSet insertResultSet = orderCommentsTbl.getInserts();

        while ( insertResultSet.next() )
        {
            // Get order comments
            int commentID = insertResultSet.getInt("comment_id");
            int orderID = insertResultSet.getInt("order_id");
            String orderComment =
insertResultSet.getString("order_comment");
            if (orderComment != null) {
                writeOrderComment(commentID,orderID,orderComment);
            }
        }
        insertResultSet.close();
    }

    public void SetDownload()
        throws SQLException, IOException
    {
        DownloadData download_d = _cc.getDownloadData();

        DownloadTableData download_td =
download_d.getDownloadTableByName( "OrderComments" );

        PreparedStatement update_ps = download_td.getUpsertPreparedStatement();
        try {
            if (_myReader == null)
```

```
                _myReader = new BufferedReader(new FileReader(
                    "C:\\MLdirect\\orderResponses.txt"));
            String commentLine = _myReader.readLine();
            while (commentLine != null) {
                // Get the next line from orderResponses.txt
                String[] responseDetails = commentLine.split("\t");

                if (responseDetails.length != 3) {
                    System.err.println("Error reading from orderResponses.txt");
                    System.err.println("Error setting direct row handling
download");
                    return;
                }
                int commentID = Integer.parseInt(responseDetails[0]);
                int orderID = Integer.parseInt(responseDetails[1]);
                String updatedComment = responseDetails[2];

                // Set an order comment response in the MobiLink download
                update_ps.setInt(1, commentID);
                update_ps.setInt(2, orderID);
                update_ps.setString(3, updatedComment);
                // Send comment response down to clients
                update_ps.executeUpdate();

                commentLine = _myReader.readLine();
            }
        }
        finally {
            update_ps.close();
        }
    }
    public void EndSync()
        throws IOException
    {
        if (_myReader != null) {
            _myReader.close();
            _myReader = null;
        }
        if (_myWriter != null) {
            _myWriter.close();
            _myWriter = null;
        }
    }
}
```

1.5.6.5.2 Complete MobiLinkOrders Code Listing (.NET)

The following is the complete MobiLinkOrders listing for .NET direct row handling.

```
using Sap.MobiLink.Script;
using System.IO;
using System.Data;
using System.Text;

public class MobiLinkOrders
{
    private DBConnectionContext _cc = null;
    private static StreamWriter _myWriter = null;
    private static StreamReader _myReader = null;
    public MobiLinkOrders(DBConnectionContext cc) {
        _cc = cc;
    }
    public void WriteOrderComment(int commentID, int orderID, string comments)
    {
        if (_myWriter == null) {
            _myWriter = new StreamWriter("c:\\MLdirect\\orderComments.txt");
        }
        _myWriter.WriteLine("{0}\t{1}\t{2}", commentID, orderID, comments);
        _myWriter.Flush();
    }
    public void GetUpload(UploadData ut)
    {
        // Get UploadedTableData for remote table called OrderComments
        UploadedTableData order_comments_table_data =
            ut.GetUploadedTableByName("OrderComments");
        // Get inserts uploaded by the MobiLink client
        IDataReader new_comment_reader =
            order_comments_table_data.GetInserts();
        while (new_comment_reader.Read()) {
            // Columns are
            // 0 - "comment_id"
            // 1 - "order_id"
            // 2 - "order_comment"
            // You can look up these values using the DataTable returned by:
            // order_comments_table_data.GetSchemaTable().
            // In this example, you just use the known column order to
            // determine the column indexes
            // Only process this insert if the order_comment column is not null
            if (!new_comment_reader.IsDBNull(2)) {
                int commentID = new_comment_reader.GetInt32(0);
                int orderID = new_comment_reader.GetInt32(1);
                string orderComment = new_comment_reader.GetString(2);
                WriteOrderComment(commentID, orderID, orderComment);
            }
        }
        new_comment_reader.Close();
    }
    private const string read_file_path = "c:\\MLdirect\\orderResponses.txt";
    public void SetDownload()
    {
        if ((_myReader == null) && !File.Exists(read_file_path)) {
            System.Console.Out.Write("There is no file to read.");
            return;
        }
        DownloadTableData comments_for_download =
            _cc.GetDownloadData().GetDownloadTableByName("OrderComments");
        // Add upserts to the set of operation that are going to be
        // applied at the remote database
        IDbCommand comments_upsert =
            comments_for_download.GetUpsertCommand();
        if (_myReader == null) {
            _myReader = new StreamReader(read_file_path);
        }
        string commentLine;
        while ((commentLine = _myReader.ReadLine()) != null) {
            // Three values are on each line separated by '\t'
            string[] responseDetails = commentLine.Split('\t');
            if (responseDetails.Length != 3) {
                throw (new SynchronizationException(
                    "Error reading from orderResponses.txt"));
            }
            int commentID = System.Int32.Parse(responseDetails[0]);
            int orderID = System.Int32.Parse(responseDetails[1]);
            string updatedComment = responseDetails[2];
            // Parameters of the correct number and type have
            // already been added so you just need to set the
            // values of the IDataParameter
            ((IDataParameter)(comments_upsert.Parameters[0])).Value =
                commentID;
            ((IDataParameter)(comments_upsert.Parameters[1])).Value =
                orderID;
            ((IDataParameter)(comments_upsert.Parameters[2])).Value =
                updatedComment;
            // Add the upsert operation
            comments_upsert.ExecuteNonQuery();
        }
    }
    public void EndSync()
    {
        if (_myWriter != null) {
            _myWriter.Close();
            _myWriter = null;
        }
        if (_myReader != null) {
            _myReader.Close();
            _myReader = null;
        }
    }
}
```

1.5.6.6 Lesson 6: Starting the MobiLink Server

In this lesson, you start the MobiLink server. Start the MobiLink server (mlsrv17) using the -c option to connect to your consolidated database. Use the -sl java or -sl dnet option to load your Java or .NET class, respectively.

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

Connect to your consolidated database and load the class on the mlsrv17 command line.

Replace `c:\MLdirect` with the location of your source files.

For Java, run the following command:

```
mlsrv17 -c "DSN=mldirect_db" -o serverOut.txt -v+ -dl -zu+ -x tcpip -sl java (-
cp c:\MLdirect)
```

For .NET, run the following command:

```
mlsrv17 -c "DSN=mldirect_db" -o serverOut.txt -v+ -dl -zu+ -x tcpip -sl dnet (-
MLAutoLoadPath=c:\MLdirect)
```

Below is a description of each MobiLink server option used in this tutorial. The options -o, -v, and -dl provide debugging and troubleshooting information. Using these logging options is appropriate in a development environment. For performance reasons, -v+ and -dl are typically not used in production.

Option	Description
-c	Precedes the connection string.
-o	Specifies the message log file `serverOut.txt`.
-v+	The -v option specifies what information is logged. Using -v+ sets maximum verbose logging.
-dl	Displays all log messages on screen.
-zu+	Adds new users automatically.
-x	Sets the communications protocol and parameters for MobiLink clients.
-sl java	Specifies a set of directories to search for class files, and forces the Java VM to load on server startup.
-sl dnet	Specifies the location of .NET assemblies and forces the CLR to load on server startup.

The MobiLink server messages window appears.

Results

The MobiLink server is started with a connection to your consolidated database and the class you created is loaded.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Direct Row Handling [page 198]

Previous task: Lesson 5: Creating a Java or .NET Class for MobiLink Direct Row Handling [page 208]

Next task: Lesson 7: Setting up Your MobiLink Client Database [page 220]

Related Information

MobiLink Server Options
-sl java mlsrv17 Option
-sl dnet mlsrv17 Option

1.5.6.7 Lesson 7: Setting up Your MobiLink Client Database

In this lesson, you use a SQL Anywhere database for your consolidated database and your MobiLink client. For tutorial purposes, your MobiLink client, consolidated database, and MobiLink server all reside on the same computer.

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

To set up the MobiLink client database, create the RemoteOrders and OrderComments tables. The RemoteOrders table corresponds to the RemoteOrders table on the consolidated database. The MobiLink server uses SQL-based scripts to synchronize remote orders. The OrderComments table is only used on client databases. The MobiLink server processes the OrderComments tables using special events.

After creating the tables, you create a synchronization user, publication, and subscription on the client database. Publications identify the tables and columns on your remote database that you want synchronized. These tables and columns are called **articles**. A synchronization subscription subscribes a MobiLink user to a publication.

Procedure

1. Create your MobiLink client database using the dbinit command line utility.

   Run the following command:

   ```
   dbinit -i -k -dba DBA,passwd remote1
   ```

   The -i and -k options omit jConnect support and Watcom SQL compatibility views, respectively.
2. Start your MobiLink client database using the dbsrv17 command line utility.

   Run the following command:

   ```
   dbsrv17 remote1
   ```
3. Run the following command to connect to your MobiLink client database from Interactive SQL:

   ```
   dbisql -c "SERVER=remote1;UID=DBA;PWD=passwd"
   ```
4. Create the RemoteOrders table by executing the following SQL statement in Interactive SQL:

   ```
   CREATE TABLE RemoteOrders (
   ```

```
    order_id            INTEGER NOT NULL,
    product_id          INTEGER NOT NULL,
    quantity            INTEGER,
    order_status        VARCHAR(10) DEFAULT 'new',
    PRIMARY KEY(order_id)
);
```

5. Create the OrderComments table by executing the following statement in Interactive SQL:

```
CREATE TABLE OrderComments (
    comment_id          INTEGER NOT NULL,
    order_id            INTEGER NOT NULL,
    order_comment       VARCHAR(255),
    PRIMARY KEY(comment_id),
    FOREIGN KEY(order_id) REFERENCES RemoteOrders(order_id)
);
```

6. Execute the following statements in Interactive SQL to create your MobiLink synchronization user, publication, and subscription:

```
CREATE SYNCHRONIZATION USER ml_sales1;
CREATE PUBLICATION order_publ (TABLE RemoteOrders, TABLE OrderComments);
CREATE SYNCHRONIZATION SUBSCRIPTION TO order_publ FOR ml_sales1
 TYPE TCPIP ADDRESS 'host=localhost';
```

> **i Note**
>
> You specify how to connect to the MobiLink server using the TYPE and ADDRESS clauses in the CREATE SYNCHRONIZATION SUBSCRIPTION statement.

You can use publications to determine what data is synchronized. In this case, you specify the entire RemoteOrders and OrderComments tables.

Results

The remote SQL Anywhere database is created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Direct Row Handling [page 198]

Previous task: Lesson 6: Starting the MobiLink Server [page 218]

Next task: Lesson 8: Synchronizing [page 222]

Related Information

MobiLink Clients
Initialization Utility (dbinit)
CREATE SYNCHRONIZATION USER Statement [MobiLink]
CREATE PUBLICATION Statement [MobiLink] [SQL Remote]
CREATE SYNCHRONIZATION SUBSCRIPTION Statement [MobiLink]

1.5.6.8 Lesson 8: Synchronizing

The dbmlsync utility initiates MobiLink synchronization for SQL Anywhere remote databases. Before starting dbmlsync, add order data and comments to your remote 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. Connect to the MobiLink client database from Interactive SQL, if you are not already connected, by running the following command:

```
dbisql -c "SERVER=remote1;UID=DBA;PWD=passwd"
```

2. Execute the following statement to add an order to the RemoteOrders table in the client database:

```
INSERT INTO RemoteOrders (order_id, product_id, quantity, order_status)
 VALUES (1,12312,10,'new');
```

3. Add a comment to the OrderComments table in the client database by executing the following statement in Interactive SQL:

```
INSERT INTO OrderComments (comment_id, order_id, order_comment)
 VALUES (1,1,'send promotional material with the order');
```

4. Execute the following statement in Interactive SQL to commit your changes:

```
COMMIT;
```

5. Run the following command:

```
dbmlsync -c "SERVER=remote1;UID=DBA;PWD=passwd" -o rem1.txt -v+
```

The following table contains a description for each dbmlsync option used in this lesson.

Option	Description
-c	Specifies the connection string.
-o	Specifies the message log file `rem1.txt`.
-v+	The -v option specifies what information is logged. Using -v+ sets maximum verbose logging.

Once you have started the MobiLink synchronization client, an output screen appears indicating that the synchronization succeeded. The SQL-based synchronization transfers rows in the client's RemoteOrders table to the RemoteOrders table in the consolidated database.

Java or .NET processing inserted your comment in `orderComments.txt`.

6. Close any SQL Anywhere MobiLink client windows.
7. Insert a response in `orderResponses.txt` to download to the remote database. This action takes place on the server side.

   Add the following text to `orderResponses.txt`. You must separate entries using the tab character. At the end of the line, press Enter.

   ```
   1    1    Promotional material shipped
   ```
8. Run synchronization using the dbmlsync client utility.

   This action takes place on the client-side.

   Run the following command:

   ```
   dbmlsync -c "SERVER=remote1;UID=DBA;PWD=passwd" -o rem1.txt -v+
   ```

   i Note

   Rows downloaded using direct row handling are not printed by the mlsrv17 -v+ option, but are printed in the remote log by the dbmlsync -v+ option.

   The MobiLink client utility appears.
9. In Interactive SQL, select from the OrderComments table to verify that the row was downloaded.

   Execute the following SQL statement:

   ```
   SELECT * FROM OrderComments;
   ```

Results

The SQL Anywhere remote database is updated and the synchronized with the consolidated database.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Direct Row Handling [page 198]

Previous task: Lesson 7: Setting up Your MobiLink Client Database [page 220]

Next task: Lesson 9: Cleaning Up [page 224]

Related Information

SQL Anywhere Clients
MobiLink SQL Anywhere Client Utility (dbmlsync) Syntax

1.5.6.9 Lesson 9: Cleaning Up

Remove the tutorial materials from your computer.

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. Close all instances of Interactive SQL.
2. Close the SQL Anywhere, MobiLink, and synchronization client windows.
3. Delete all tutorial-related ODBC data sources:
    a. Start the ODBC Administrator.

       Run the following command:

       ```
       odbcad32
       ```
    b. Remove the **`mldirect_db`** data source.
4. Delete the consolidated and remote databases:
    a. Navigate to the directory containing your consolidated and remote databases.
    b. Delete `MLconsolidated.db`, `MLconsolidated.log`, `remote1.db`, and `remote1.log`.

Results

The tutorial materials are removed from your computer.

Task overview: Tutorial: Using Direct Row Handling [page 198]

Previous task: Lesson 8: Synchronizing [page 222]

1.5.7 Tutorial: Synchronizing with Microsoft Excel

You can use direct row handling to communicate remote data to any central data source, application, or web service.

Prerequisites

You require:

- Familiarity with Java
- Familiarity with Microsoft Excel
- Basic knowledge of MobiLink event scripts

The following software is required:

- SQL Anywhere 17
- Java Software Development Kit
- Microsoft Office Excel 2007 or later

You must have the following roles and privileges on the consolidated database:

- SYS_AUTH_RESOURCE_ROLE compatibility role
- MONITOR system privilege

You must have the following roles and privileges on the remote database:

- SYS_REPLICATION_ADMIN_ROLE system role
- SYS_RUN_REPLICATION_ROLE system role

Context

This tutorial guides you through the basic steps for using direct row handling to synchronize data in a Microsoft Excel spreadsheet with MobiLink clients. It shows you how to implement MobiLink direct row handling using a Java implementation as an example so that you can use a data source other than a supported consolidated database.

This tutorial shows you how to:

- Use the MobiLink server API for Java
- Create methods for MobiLink direct row handling
- Access data from a Microsoft Excel worksheet using Java

1. Lesson 1: Setting up a Microsoft Excel Worksheet [page 227]
   In this lesson, you create a Microsoft Excel worksheet and use the Microsoft Excel Driver to define an ODBC data source. The Microsoft Excel worksheet stores product information.
2. Lesson 2: Setting up Your MobiLink Consolidated Database [page 228]
   In this lesson, you create a database and define an ODBC data source.
3. Lesson 3: Creating a Table in Your MobiLink Consolidated Database [page 230]
   In this lesson, you create the RemoteOrders table in the MobiLink consolidated database.
4. Lesson 4: Adding Synchronization Scripts [page 232]
   In this lesson you add scripts to your consolidated database for SQL row handling and direct row handling.
5. Lesson 5: Creating a Java Class for MobiLink Direct Row Handling [page 234]
   In this lesson, you use direct row handling to process rows in the OrderComments table in your client database.
6. Lesson 6: Starting the MobiLink Server [page 240]
   In this lesson, you start the MobiLink server (mlsrv17) using the -c option to connect to your consolidated database, and the -sl java option to load your Java class.
7. Lesson 7: Setting up Your MobiLink Client Database [page 242]
   In this lesson, you use a SQL Anywhere database for your consolidated database and your MobiLink client.
8. Lesson 8: Synchronizing [page 244]
   The dbmlsync utility initiates MobiLink synchronization for SQL Anywhere remote databases. Before starting dbmlsync, add order data and comments to your remote database.
9. Lesson 9: Cleaning Up [page 246]
   Remove the tutorial materials from your computer.

Related Information

MobiLink Synchronization [page 4]

Synchronization Techniques
Direct Row Handling

SAP SQL Anywhere Forum

1.5.7.1 Lesson 1: Setting up a Microsoft Excel Worksheet

In this lesson, you create a Microsoft Excel worksheet and use the Microsoft Excel Driver to define an ODBC data source. The Microsoft Excel worksheet stores product information.

Prerequisites

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

The Microsoft Excel driver is a 32-bit driver, so the 32-bit version of the ODBC Data Source Administrator is required for this tutorial.

Procedure

1. Create a working directory called `c:\MLobjexcel` for server-side components.
2. Open Microsoft Excel and create a new workbook.
3. In the default worksheet, add the following contents under the respective **`A`**, **`B`**, **`C`** column headers:

comment_id	order_id	order_comment
2	1	Promotional material shipped
3	1	More information about material required

4. Change the default worksheet name **`Sheet1`** to **`order_sheet`**.
   a. Double-click the **`Sheet1`** tab.
   b. Type **`order_sheet`**.
5. Save the Microsoft Excel workbook. Save the workbook as `order_central.xlsx` in the `c:\MLobjexcel` working directory.
6. Use the Microsoft Excel Driver to create an ODBC data source:
   a. Click *Start* › *Programs* › *SQL Anywhere 17* › *Administration Tools* › *ODBC Data Source Administrator*.
   b. Click the *User DSN* tab.
   c. Click *Add*.
   d. Click *Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)*.
   e. Click *Finish*.
   f. In the *Data Source Name* field, type **`excel_datasource`**.
   g. Click *Select Workbook* and browse to `c:\MLobjexcel\order_central.xlsx`, the file containing your worksheet.
   h. Clear the *Read Only* option.
   i. Click *OK* on all open ODBC Data Source Administrator windows.
7. If there is no Microsoft Excel Driver, execute `%windir%\SysWOW64\odbcad32.exe`. See Unable to create DSN for Microsoft Office System Driver on 64-bit versions of Windows .

Results

A Microsoft Excel worksheet and an ODBC data source are created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Synchronizing with Microsoft Excel [page 225]

Next task: Lesson 2: Setting up Your MobiLink Consolidated Database [page 228]

1.5.7.2 Lesson 2: Setting up Your MobiLink Consolidated Database

In this lesson, you create a database and define an ODBC data source.

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

Your MobiLink consolidated database is a central repository of data and includes MobiLink system tables and stored procedures used to manage the synchronization process. With direct row handling, you synchronize with a data source other than a consolidated database, but you still need a consolidated database to maintain information used by the MobiLink server.

> i Note
>
> If you already have a MobiLink consolidated database set up with MobiLink system objects and a DSN, you can skip this lesson.

Procedure

1. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Central*.
2. Click *Tools* > *SQL Anywhere 17* > *Create Database*.
3. Click *Next*.
4. Leave the default of *Create a database on this computer*, and then click *Next*.
5. In the *Save the main database file to the following file* field, type the file name and path for the database. For example, `c:\MLobjexcel\MLconsolidated.db`.
6. Follow the remaining instructions in the *Create Database Wizard* and accept the default values. If prompted to specify a user ID and password for the DBA user, enter **`DBA`** and **`passwd`**, respectively.

   On the *Connect To The Database* page, clear the *Stop the database after last disconnect* option.
7. Click *Finish*.

   The MLconsolidated database appears in SQL Central.
8. In SQL Central, click *Tools* > *SQL Anywhere 17* > *Open ODBC Administrator*.
9. Click the *User DSN* tab, and then click *Add*.
10. In the *Create New Data Source* window, click *SQL Anywhere 17*, and then click *Finish*.
11. Perform the following tasks in the *ODBC Configuration For SQL Anywhere* window:
    a. Click the *ODBC* tab.
    b. In the *Data source name* field, type **`mlexcel_db`**.
    c. Click the *Login* tab.
    d. In the *User ID* field, type **`DBA`**.
    e. In the *Password* field, type **`passwd`**.
    f. From the *Action* dropdown list, click *Connect to a running database on this computer*.
    g. In the *Server name* field, type **`MLconsolidated`**.
    h. Click *OK*.
12. Close ODBC data source administrator.

    Click *OK* on the *ODBC Data Source Administrator* window.

Results

A consolidated database and an ODBC data source for the consolidated database are created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Synchronizing with Microsoft Excel [page 225]

Previous task: Lesson 1: Setting up a Microsoft Excel Worksheet [page 227]

Next task: Lesson 3: Creating a Table in Your MobiLink Consolidated Database [page 230]

Related Information

MobiLink Consolidated Databases
Initialization Utility (dbinit)
CREATE TABLE Statement

1.5.7.3 Lesson 3: Creating a Table in Your MobiLink Consolidated Database

In this lesson, you create the RemoteOrders table in the MobiLink consolidated 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

The RemoteOrders table you create in this lesson contains the following columns:

order_id

A unique identifier for orders.

product_id

A unique identifier for products.

quantity

The number of items sold.

order_status

The order status.

last_modified

The last modification date of a row. You use this column for timestamp-based downloads, a common technique used to filter rows for efficient synchronization.

Procedure

1. Connect to your database from Interactive SQL.

   You can start Interactive SQL from SQL Central or at a command prompt.

   - To start Interactive SQL from SQL Central, right-click the **`MLconsolidated - DBA`** database and click *Open Interactive SQL*.
   - To start Interactive SQL at a command prompt, run the following command:

     ```
     dbisql -c "DSN=mlexcel_db"
     ```

2. Execute the following SQL statements in Interactive SQL to create the RemoteOrders table.

   ```
   CREATE TABLE RemoteOrders (
       order_id            INTEGER NOT NULL,
       product_id          INTEGER NOT NULL,
       quantity            INTEGER,
       order_status        VARCHAR(10) DEFAULT 'new',
       last_modified       TIMESTAMP DEFAULT CURRENT TIMESTAMP,
       PRIMARY KEY(order_id)
   );
   ```

   Interactive SQL creates the RemoteOrders table in your consolidated database.
3. Execute the following SQL statement in Interactive SQL to create MobiLink system tables and stored procedures.

   Replace *C:\Program Files\SQL Anywhere 17*\ with the location of your SQL Anywhere 17 installation.

   ```
   READ "C:\Program Files\SQL Anywhere 17\MobiLink\setup\syncsa.sql";
   ```

   Interactive SQL applies `syncsa.sql` to your consolidated database. Running `syncsa.sql` creates a series of system tables and stored procedures prefaced with **ml_**. The MobiLink server works with these tables and stored procedures in the synchronization process.

Results

The RemoteOrders table is created in the consolidated database.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Synchronizing with Microsoft Excel [page 225]

Previous task: Lesson 2: Setting up Your MobiLink Consolidated Database [page 228]

Next task: Lesson 4: Adding Synchronization Scripts [page 232]

1.5.7.4 Lesson 4: Adding Synchronization Scripts

In this lesson you add scripts to your consolidated database for SQL row handling and direct row handling.

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 row handling allows you to synchronize remote data with tables in your MobiLink consolidated database. SQL-based scripts define:

- How data that is uploaded from a MobiLink client is to be applied to the consolidated database.
- What data should be downloaded from the consolidated database.

In this lesson, you write synchronization scripts for the following SQL-based upload and download events:

upload_insert

This event defines how new orders inserted in a client database should be applied to the consolidated database.

download_cursor

This event defines the orders that should be downloaded to remote clients.

download_delete_cursor

This event is required when using synchronization scripts that are not upload-only. You set the MobiLink server to ignore this event for the purpose of this tutorial.

You use direct row handling to add special processing to a SQL-based synchronization system. In this procedure you register method names corresponding to the handle_UploadData, handle_DownloadData, download_cursor, and download_delete_cursor events. You create your own Java class in a later lesson.

Procedure

1. Connect to your consolidated database from Interactive SQL if you are not already connected.

   Run the following command:

   ```
   dbisql -c "DSN=mlexcel_db"
   ```

2. Use the ml_add_table_script stored procedure to add SQL-based table scripts for the upload_insert, download_cursor and download_delete_cursor events.

Execute the following SQL statements in Interactive SQL. The upload_insert script inserts the uploaded order_id, product_id, quantity, and order_status into the MobiLink consolidated database. The download_cursor script uses timestamp-based filtering to download updated rows to remote clients.

```
CALL ml_add_table_script( 'default', 'RemoteOrders',
    'upload_insert',
    'INSERT INTO RemoteOrders( order_id, product_id, quantity, order_status)
    VALUES( {ml r.order_id}, {ml r.product_id}, {ml r.quantity}, {ml
r.order_status} )' );

CALL ml_add_table_script( 'default', 'RemoteOrders',
    'download_cursor',
    'SELECT order_id, product_id, quantity, order_status
    FROM RemoteOrders WHERE last_modified >= {ml s.last_table_download}');
CALL ml_add_table_script( 'default', 'RemoteOrders',
    'download_delete_cursor', '--{ml_ignore}');
COMMIT
```

3. Register Java methods for the handle_UploadData and handle_DownloadData events.

   Execute the following SQL statements in Interactive SQL:

```
CALL ml_add_java_connection_script( 'default',
    'handle_UploadData',
    'MobiLinkOrders.GetUpload' );

CALL ml_add_java_connection_script( 'default',
    'handle_DownloadData',
    'MobiLinkOrders.SetDownload' );
```

   Interactive SQL registers the GetUpload and SetDownload methods for the handle_UploadData and handle_DownloadData events, respectively. You create these methods in an upcoming lesson.
4. Register the download_cursor and download_delete_cursor events.

   Run the following SQL script in Interactive SQL:

```
CALL ml_add_table_script( 'default', 'OrderComments',
    'download_cursor', '--{ml_ignore}');
CALL ml_add_table_script( 'default', 'OrderComments',
    'download_delete_cursor', '--{ml_ignore}');
```

   The download_cursor and download_delete_cursor events must be registered for the OrderComments table when using scripts because the synchronization is bi-directional and not upload-only.
5. Commit your changes.

   Execute the following SQL statement in Interactive SQL:

```
COMMIT;
```

Results

The upload_insert, download_cursor and download_delete_cursor events are added to the database. The method names corresponding to the handle_UploadData, handle_DownloadData, download_cursor, and download_delete_cursor events are registered.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Synchronizing with Microsoft Excel [page 225]

Previous task: Lesson 3: Creating a Table in Your MobiLink Consolidated Database [page 230]

Next task: Lesson 5: Creating a Java Class for MobiLink Direct Row Handling [page 234]

Related Information

Overview of MobiLink Events
Scripts Required for Synchronization
Script Additions and Deletions
Scripts to Upload Rows
Scripts to Download Rows
Direct Row Handling
Direct Uploads
Direct Downloads
Partitioned Rows Among Remote Databases
Implementing Timestamp-based Downloads
upload_insert Table Event
upload_update Table Event
upload_delete Table Event
download_cursor Table Event
download_delete_cursor Table Event

1.5.7.5 Lesson 5: Creating a Java Class for MobiLink Direct Row Handling

In this lesson, you use direct row handling to process rows in the OrderComments table in your client 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

In the lesson, you add the following methods for direct row handling:

GetUpload

You use this method for the handle_UploadData event. GetUpload writes uploaded comments to the Microsoft Excel worksheet `order_central.xlsx`.

SetDownload

You use this method for the handle_DownloadData event. SetDownload retrieves the data stored in the Microsoft Excel worksheet `order_central.xlsx` and sends it to remote clients.

The following procedure shows you how to create a Java class including your methods for processing. For a complete listing, see the MobiLinkOrders code listing for Java.

Procedure

1. Start writing a new class named MobiLinkOrders.

   Write the following code:

   ```
   import com.sap.ml.script.*;
   import java.io.*;
   import java.sql.*;
   public class MobiLinkOrders {
   ```

2. Declare a class-level DBConnectionContext instance.

   Append the following code:

   ```
       // Class level DBConnectionContext
       DBConnectionContext _cc;
   ```

   The MobiLink server passes a DBConnectionContext instance to your class constructor. DBConnectionContext encapsulates information about the current connection with the MobiLink consolidated database.

3. Create your class constructor. Your class constructor sets your class-level DBConnectionContext instance.

   Append the following code:

   ```
       public MobiLinkOrders( DBConnectionContext cc )
           throws IOException, FileNotFoundException {
           // Declare a class-level DBConnectionContext
           _cc = cc;
       }
   ```

4. Write the GetUpload method.

   The GetUpload method obtains an UploadedTableData class instance representing the OrderComments table. The OrderComments table contains special comments made by remote sales employees. You create this table in a later lesson.

   The UploadedTableData getInserts method returns a result set for new order comments.

Append the following code:

```
    //  Method for the handle_UploadData synchronization event
    public void GetUpload( UploadData ut )
        throws SQLException, IOException {
        // Get an UploadedTableData for OrderComments
        UploadedTableData orderCommentsTbl =
ut.getUploadedTableByName("OrderComments");
        // Get inserts uploaded by the MobiLink client
        ResultSet insertResultSet = orderCommentsTbl.getInserts();

        try {
            // Connect to the excel worksheet through ODBC
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection con =
DriverManager.getConnection( "jdbc:odbc:excel_datasource" );
            while( insertResultSet.next() ) {
                // Get order comments
                int _commentID = insertResultSet.getInt("comment_id");
                int _orderID = insertResultSet.getInt("order_id");
                String _specialComments =
insertResultSet.getString("order_comment");
                // Execute an insert statement to add the order comment to
the worksheet
                PreparedStatement st = con.prepareStatement("INSERT INTO
[order_sheet$]"
                    + "(order_id, comment_id, order_comment) VALUES
(?,?,?)" );
                st.setString( 1, Integer.toString(_orderID) );
                st.setString( 2, Integer.toString(_commentID) );
                st.setString( 3, _specialComments );
                st.executeUpdate();
                st.close();
            }
            con.close();
        } catch(Exception ex) {
            System.err.print("Exception: ");
            System.err.println(ex.getMessage());
        } finally {
        insertResultSet.close();
    }
    }
```

5. Write the SetDownload method:
    a. Obtain a class instance representing the OrderComments table.

       Use the DBConnectionContext getDownloadData method to obtain a DownloadData instance. Use the DownloadData getDownloadTableByName method to return a DownloadTableData instance for the OrderComments table.

       Append the following code:

```
    public void SetDownload() throws SQLException, IOException {
        DownloadData download_d = _cc.getDownloadData();
        DownloadTableData download_td =
download_d.getDownloadTableByName( "OrderComments" );
```

> **i Note**
>
> You create this table on the remote database when you set up your MobiLink client database.

    b. Obtain a prepared statement or IDbCommand that allows you to add insert or update operations to the download.

Use the DownloadTableData getUpsertPreparedStatement method to return a java.sql.PreparedStatement instance.

Append the following code:

```
        // Prepared statement to compile upserts (inserts or updates).
        PreparedStatement download_upserts =
download_td.getUpsertPreparedStatement();
```

c. Set the download data for each row.

The following code traverses through the `order_central.xlsx` worksheet and adds data to the MobiLink download.

Append the following code:

```

        try {
            // Connect to the excel worksheet through ODBC
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection con =
DriverManager.getConnection( "jdbc:odbc:excel_datasource" );
            // Retrieve all the rows in the worksheet
            Statement st = con.createStatement();
            ResultSet Excel_rs = st.executeQuery( "select * from
[order_sheet$]" );
            while (Excel_rs.next()) {
                // Retrieve the row data
                int Excel_comment_id = Excel_rs.getInt(1);
                int Excel_order_id = Excel_rs.getInt(2);
                String Excel_comment = Excel_rs.getString(3);
                // Add the Excel data to the MobiLink download.
                download_upserts.setInt( 1, Excel_comment_id );
                download_upserts.setInt( 2, Excel_order_id );
                download_upserts.setString( 3, Excel_comment );
                download_upserts.executeUpdate();
            }
            // Close the excel result set, statement, and connection.
            Excel_rs.close();
            st.close();
            con.close();
        } catch (Exception ex) {
            System.err.print("Exception: ");
            System.err.println(ex.getMessage());
        }
```

d. Close the prepared statement used for adding insert or update operations to the download, and end the method and the class.

Append the following code:

```
        finally {
        download_upserts.close();
    }
    }
}
```

6. Save your Java code as `MobiLinkOrders.java` in your working directory `c:\MLobjexcel`.

   To verify the code in `MobiLinkOrders.java`, see the MobiLinkOrders code listing for Java.

7. Compile your class file.

   a. Navigate to the directory containing your Java source files.
   b. Compile MobiLinkOrders that refer to the MobiLink server API library for Java.

You need to reference `mlscript.jar`, located in *%SQLANY17%*`\Java`.

Run the following command, replacing *C:\Program Files\SQL Anywhere 17* with your SQL Anywhere 17 directory:

```
javac -classpath "C:\Program Files\SQL Anywhere 17\java\mlscript.jar"
MobiLinkOrders.java
```

Results

Rows in the OrderComments table in your client database are updated.

Next Steps

Proceed to the next lesson.

In this section:

Complete MobiLinkOrders Code Listing (Java) [page 238]
The following listing shows the complete Java MobiLinkOrders class code used for this tutorial.

Task overview: Tutorial: Synchronizing with Microsoft Excel [page 225]

Previous task: Lesson 4: Adding Synchronization Scripts [page 232]

Next task: Lesson 6: Starting the MobiLink Server [page 240]

Related Information

Direct Row Handling
Synchronization Script Writing in Java

1.5.7.5.1 Complete MobiLinkOrders Code Listing (Java)

The following listing shows the complete Java MobiLinkOrders class code used for this tutorial.

```
import com.sap.ml.script.*;
    import java.io.*;
    import java.sql.*;

    public class MobiLinkOrders {
```

```
    // Class level DBConnectionContext
    DBConnectionContext _cc;

    public MobiLinkOrders( DBConnectionContext cc )
        throws IOException, FileNotFoundException {
        // Declare a class-level DBConnectionContext
        _cc = cc;
    }
    //  Method for the handle_UploadData synchronization event
    public void GetUpload( UploadData ut )
        throws SQLException, IOException {
        // Get an UploadedTableData for OrderComments
        UploadedTableData orderCommentsTbl =
ut.getUploadedTableByName("OrderComments");
        // Get inserts uploaded by the MobiLink client
        ResultSet insertResultSet = orderCommentsTbl.getInserts();

        try {
            // Connect to the excel worksheet through ODBC
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection con =
DriverManager.getConnection( "jdbc:odbc:excel_datasource" );
            while( insertResultSet.next() ) {
                // Get order comments
                int _commentID = insertResultSet.getInt("comment_id");
                int _orderID = insertResultSet.getInt("order_id");
                String _specialComments =
insertResultSet.getString("order_comment");
                // Execute an insert statement to add the order comment to the
worksheet
                PreparedStatement st = con.prepareStatement("INSERT INTO
[order_sheet$]"
                    + "(order_id, comment_id, order_comment) VALUES (?,?,?)" );
                st.setString( 1, Integer.toString(_orderID) );
                st.setString( 2, Integer.toString(_commentID) );
                st.setString( 3, _specialComments );
                st.executeUpdate();
                st.close();
            }
            con.close();
        } catch(Exception ex) {
            System.err.print("Exception: ");
            System.err.println(ex.getMessage());
        } finally {
        insertResultSet.close();
    }
    }

    public void SetDownload() throws SQLException, IOException {
        DownloadData download_d = _cc.getDownloadData();
        DownloadTableData download_td =
download_d.getDownloadTableByName( "OrderComments" );

        // Prepared statement to compile upserts (inserts or updates).
        PreparedStatement download_upserts =
download_td.getUpsertPreparedStatement();

        try {
            // Connect to the excel worksheet through ODBC
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection con =
DriverManager.getConnection( "jdbc:odbc:excel_datasource" );
            // Retrieve all the rows in the worksheet
            Statement st = con.createStatement();
            ResultSet Excel_rs = st.executeQuery( "select * from [order_sheet
$]" );
            while (Excel_rs.next()) {
                // Retrieve the row data
                int Excel_comment_id = Excel_rs.getInt(1);
                int Excel_order_id = Excel_rs.getInt(2);
                String Excel_comment = Excel_rs.getString(3);
                // Add the Excel data to the MobiLink download.
                download_upserts.setInt( 1, Excel_comment_id );
                download_upserts.setInt( 2, Excel_order_id );
                download_upserts.setString( 3, Excel_comment );
                download_upserts.executeUpdate();
            }
            // Close the excel result set, statement, and connection.
            Excel_rs.close();
            st.close();
            con.close();
        } catch (Exception ex) {
            System.err.print("Exception: ");
            System.err.println(ex.getMessage());
        } finally {
        download_upserts.close();
    }
    }
}
```

1.5.7.6 Lesson 6: Starting the MobiLink Server

In this lesson, you start the MobiLink server (mlsrv17) using the -c option to connect to your consolidated database, and the -sl java option to load your Java class.

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

Connect to your consolidated database and load the class on the mlsrv17 command line.

Ensure your consolidated database is running and then run the following command. Replace `c:\MLobjexcel` with the location of your Java source files.

```
mlsrv17 -c "DSN=mlexcel_db" -o serverOut.txt -v+ -dl -zu+ -x tcpip  -sl java (-
cp c:\MLobjexcel)
```

The MobiLink server messages window appears.

Below is a description of each MobiLink server option used in this tutorial. The options -o, -v, and -dl provide debugging and troubleshooting information. Using these logging options is appropriate in a development environment. For performance reasons, -v+ and -dl are typically not used in production.

Option	Description
-c	Precedes the connection string.
-o	Specifies the message log file `serverOut.txt`.
-v+	The -v option specifies what information is logged. Using -v+ sets maximum verbose logging.
-dl	Displays all log messages on screen.
-zu+	Adds new users automatically.
-x	Sets the communications protocol and parameters for Mobi-Link clients.
-sl java	Specifies a set of directories to search for class files, and forces the Java VM to load on server startup.

The MobiLink server messages window appears.

Results

The MobiLink server is started and ready for direct row handling.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Synchronizing with Microsoft Excel [page 225]

Previous task: Lesson 5: Creating a Java Class for MobiLink Direct Row Handling [page 234]

Next task: Lesson 7: Setting up Your MobiLink Client Database [page 242]

Related Information

MobiLink Server Options
-sl java mlsrv17 Option

1.5.7.7 Lesson 7: Setting up Your MobiLink Client Database

In this lesson, you use a SQL Anywhere database for your consolidated database and your MobiLink client.

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

For tutorial purposes, your MobiLink client, consolidated database, and MobiLink server all reside on the same computer.

To set up the MobiLink client database, you create the RemoteOrders and OrderComments tables. The RemoteOrders table corresponds to the RemoteOrders table on the consolidated database. The MobiLink server uses SQL-based scripts to synchronize remote orders. The OrderComments table is only used on client databases. The MobiLink server processes the OrderComments tables using special events.

Procedure

1. Create your MobiLink client database using the dbinit command line utility.

   Run the following command:

   ```
   dbinit -i -k -dba DBA,passwd remote1
   ```

   The -i and -k options omit jConnect support and Watcom SQL compatibility views, respectively.
2. Start your MobiLink client database using the dbsrv17 command line utility.

   Run the following command:

   ```
   dbsrv17 remote1
   ```
3. Connect to your MobiLink client database using Interactive SQL.

   Run the following command:

   ```
   dbisql -c "SERVER=remote1;UID=DBA;PWD=passwd"
   ```
4. Create the RemoteOrders table.

   Execute the following SQL statement in Interactive SQL:

   ```
   CREATE TABLE RemoteOrders (
       order_id           INTEGER NOT NULL,
   ```

```
    product_id          INTEGER NOT NULL,
    quantity            INTEGER,
    order_status        VARCHAR(10) DEFAULT 'new',
    PRIMARY KEY(order_id)
);
```

5. Create the OrderComments table.

   Execute the following SQL statement in Interactive SQL:

```
CREATE TABLE OrderComments (
    comment_id          INTEGER NOT NULL,
    order_id            INTEGER NOT NULL,
    order_comment       VARCHAR(255),
    PRIMARY KEY(comment_id),
    FOREIGN KEY(order_id) REFERENCES RemoteOrders(order_id)
);
```

6. Create your MobiLink synchronization user, publication, and subscription.

   Execute the following SQL statement in Interactive SQL:

```
CREATE SYNCHRONIZATION USER ml_sales1;
CREATE PUBLICATION order_publ (TABLE RemoteOrders, TABLE OrderComments);
CREATE SYNCHRONIZATION SUBSCRIPTION TO order_publ FOR ml_sales1
 TYPE TCPIP ADDRESS 'host=localhost';
```

   i Note

   You specify how to connect to the MobiLink server using the TYPE and ADDRESS clauses in the CREATE SYNCHRONIZATION SUBSCRIPTION statement.

   You can use publications to determine what data is synchronized. In this case you specify the entire RemoteOrders and OrderComments tables.

Results

The SQL Anywhere client database is created and prepared for synchronization.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Synchronizing with Microsoft Excel [page 225]

Previous task: Lesson 6: Starting the MobiLink Server [page 240]

Next task: Lesson 8: Synchronizing [page 244]

Related Information

MobiLink Clients
Initialization Utility (dbinit)
CREATE SYNCHRONIZATION USER Statement [MobiLink]
CREATE PUBLICATION Statement [MobiLink] [SQL Remote]
CREATE SYNCHRONIZATION SUBSCRIPTION Statement [MobiLink]

1.5.7.8 Lesson 8: Synchronizing

The dbmlsync utility initiates MobiLink synchronization for SQL Anywhere remote databases. Before starting dbmlsync, add order data and comments to your remote 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. Connect to the MobiLink client database from Interactive SQL if you are not already connected.

   Run the following command:

   ```
   dbisql -c "SERVER=remote1;UID=DBA;PWD=passwd"
   ```
2. Add an order to the RemoteOrders table in the client database.

   Execute the following SQL statement in Interactive SQL:

   ```
   INSERT INTO RemoteOrders (order_id, product_id, quantity, order_status)
    VALUES (1,12312,10,'new');
   ```
3. Add a comment to the OrderComments table in the client database.

   Execute the following SQL statement in Interactive SQL:

   ```
   INSERT INTO OrderComments (comment_id, order_id, order_comment)
    VALUES (1,1,'send promotional material with the order');
   ```
4. Commit your changes.

   Execute the following SQL statement in Interactive SQL:

   ```
   COMMIT;
   ```

5. Run the following command at a command prompt:

```
dbmlsync -c "SERVER=remote1;UID=DBA;PWD=passwd" -o rem1.txt -v+
```

   The following table contains a description for each dbmlsync option used:

Option	Description
-c	Specifies the connection string.
-o	Specifies the message log file `rem1.txt`.
-v+	The -v option specifies what information is logged. Using -v+ sets maximum verbose logging.

   Once you have started the MobiLink synchronization client, an output screen appears indicating that the synchronization succeeded. SQL-based synchronization transferred rows in the client RemoteOrders table to the RemoteOrders table in the consolidated database.

   Java processing inserted your comment in the `order_central.xlsx` worksheet. The information stored in the `order_central.xlsx` worksheet is downloaded to the client.

6. In Interactive SQL, select from the OrderComments table to verify that the row was downloaded.

   Execute the following SQL statement in Interactive SQL:

```
SELECT * FROM OrderComments;
```

> i Note
>
> Rows downloaded using direct row handling are not printed by the mlsrv17 -v+ option, but are printed in the remote log by the dbmlsync -v+ option.

Results

The remote database is updated with order data and comments, and the remote database and consolidated database are synchronized.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Synchronizing with Microsoft Excel [page 225]

Previous task: Lesson 7: Setting up Your MobiLink Client Database [page 242]

Next task: Lesson 9: Cleaning Up [page 246]

Related Information

SQL Anywhere Clients
MobiLink SQL Anywhere Client Utility (dbmlsync) Syntax

1.5.7.9 Lesson 9: Cleaning Up

Remove the tutorial materials from your computer.

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. Close all instances of the following applications:
   - Interactive SQL
   - Microsoft Excel
2. Delete the Microsoft Excel workbook `order_central.xlsx`.
3. Close the SQL Anywhere, MobiLink, and synchronization client windows.
4. Delete all tutorial-related ODBC data sources.
   a. Start the ODBC Data Source Administrator.

      Run the following command:

      ```
      odbcad32
      ```
   b. Remove the **`excel_datasource`** and **`mlexcel_db`** data sources.
5. Delete the consolidated and remote databases.
   a. Navigate to the directory containing your consolidated and remote databases.
   b. Delete `MLconsolidated.db`, `MLconsolidated.log`, `remote1.db`, and `remote1.log`.

Results

The tutorial materials are removed from your computer.

Task overview: Tutorial: Synchronizing with Microsoft Excel [page 225]

Previous task: Lesson 8: Synchronizing [page 244]

1.5.8 Tutorial: Synchronizing with XML

This tutorial shows you how to synchronize data in an XML file to remote clients.

Prerequisites

You require:

- Familiarity with Java
- Familiarity with XML
- Familiarity with XML DOM
- Basic knowledge of MobiLink event scripts

The following software is required:

- SQL Anywhere 17
- Java Software Development Kit
- XML DOM library

You must have the following roles and privileges on the consolidated database:

- SYS_AUTH_RESOURCE_ROLE compatibility role
- MONITOR system privilege

You must have the following roles and privileges on the remote database:

- SYS_REPLICATION_ADMIN_ROLE system role
- SYS_RUN_REPLICATION_ROLE system role

Context

You can use direct row handling to communicate remote data to any central data source, application, or web service.

In this tutorial, you implement MobiLink direct row handling so that you can use a data source other than a supported consolidated data source. This tutorial uses a Java implementation as an example.

This tutorial shows you how to:

- Use the MobiLink server Java API for Java
- Create methods for MobiLink direct row handling

1. Lesson 1: Setting up an XML Data Source [page 248]
   In this lesson, you create an XML file to store order information.
2. Lesson 2: Setting up Your MobiLink Consolidated Database [page 249]
   In this lesson, you create a database and define an ODBC data source.
3. Lesson 3: Creating a Table in Your MobiLink Consolidated Database [page 251]
   In this lesson, you create a RemoteOrders table in the MobiLink consolidated database.
4. Lesson 4: Adding Synchronization Scripts [page 253]
   In this lesson, you add scripts to your consolidated database for SQL row handling and direct row handling.
5. Lesson 5: Creating a Java Class for MobiLink Direct Row Handling [page 256]
   In this lesson, you use direct row handling to process rows in the OrderComments table in your client database. You add the GetUpload methods for direct row handling for the handle_UploadData event. GetUpload writes uploaded comments to the XML file.
6. Lesson 6: Starting the MobiLink Server [page 263]
   In this lesson, you start the MobiLink server. You start the MobiLink server (mlsrv17) using the -c option to connect to your consolidated database, and the -sl java option to load your Java class.
7. Lesson 7: Setting up Your MobiLink Client Database [page 264]
   In this lesson, you use a SQL Anywhere database for your consolidated database and your MobiLink client. For tutorial purposes your MobiLink client, consolidated database, and MobiLink server all reside on the same computer.
8. Lesson 8: Synchronizing [page 266]
   In this lesson you use the dbmlsync utility to initiate MobiLink synchronization. Before starting dbmlsync, add order data and comments to your remote database.
9. Lesson 9: Cleaning Up [page 268]
   Remove the tutorials materials from your computer.

Related Information

MobiLink Synchronization [page 4]
Synchronization Techniques
Direct Row Handling
SAP SQL Anywhere Forum

1.5.8.1 Lesson 1: Setting up an XML Data Source

In this lesson, you create an XML file to store order information.

Prerequisites

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

Procedure

1. Create an XML file with the following contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<orders></orders>
```

2. Save the XML file. This tutorial assumes `c:\MLobjxml` as the working directory for server-side components. Save the XML file as `order_comments.xml` in this directory.

Results

The XML file is created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Synchronizing with XML [page 247]

Next task: Lesson 2: Setting up Your MobiLink Consolidated Database [page 249]

1.5.8.2 Lesson 2: Setting up Your MobiLink Consolidated Database

In this lesson, you create a database and define an ODBC data source.

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

Your MobiLink consolidated database is a central repository of data and includes MobiLink system tables and stored procedures used to manage the synchronization process. With direct row handling, you synchronize with a data source other than a consolidated database, but you still need a consolidated database to maintain information used by the MobiLink server.

> **i Note**
>
> If you already have a MobiLink consolidated database set up with MobiLink system objects and a DSN, you can skip this lesson.

Procedure

1. Start SQL Central.

   Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Central*.
2. Click *Tools* > *SQL Anywhere 17* > *Create Database*.
3. Click *Next*.
4. Leave the default of *Create a database on this computer* and click *Next*.
5. In the *Save the main database file to the following file* field, type the file name and path for the database. For example, `c:\MLobjxml\MLconsolidated.db`. Click *Next*.
6. Follow the remaining instructions in the *Create Database Wizard* and accept the default values. If prompted to specify a user ID and password for the DBA user, enter **`DBA`** and **`passwd`**, respectively.

   On the *Connect To The Database* page, clear the *Stop the database after last disconnect* option.
7. Click *Finish*.

   The MLconsolidated database appears in SQL Central.
8. Click *Close* on the *Creating Database* window if the window did not close automatically.
9. Click *Tools* > *SQL Anywhere 17* > *Open ODBC Administrator*.
10. Click the *User DSN* tab and click *Add*.
11. In the *Create New Data Source* window, click *SQL Anywhere 17*, and then click *Finish*.
12. Perform the following tasks in the *ODBC Configuration For SQL Anywhere* window:
    a. Click the *ODBC* tab.
    b. In the *Data source name* field, type **`mlxml_db`**.
    c. Click the *Login* tab.
    d. In the *User ID* field, type **`DBA`**.
    e. In the *Password* field, type **`passwd`**.
    f. In the *Server name* field, type **`MLconsolidated`**.
    g. Click *OK*.
13. Close ODBC data source administrator.

    Click *OK* on the *ODBC Data Source Administrator* window.

Results

A database is created and ODBC data source is defined.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Synchronizing with XML [page 247]

Previous task: Lesson 1: Setting up an XML Data Source [page 248]

Next task: Lesson 3: Creating a Table in Your MobiLink Consolidated Database [page 251]

Related Information

MobiLink Consolidated Databases
Initialization Utility (dbinit)

1.5.8.3 Lesson 3: Creating a Table in Your MobiLink Consolidated Database

In this lesson, you create a RemoteOrders table in the MobiLink consolidated 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

The RemoteOrders table contains the following columns:

order_id

A unique identifier for orders.

product_id

A unique identifier for products.

quantity

The number of items sold.

order_status

The order status.

last_modified

The last modification date of a row. You use this column for timestamp-based downloads, a common technique used to filter rows for efficient synchronization.

Procedure

1. Connect to your database using Interactive SQL.

   You can start Interactive SQL from SQL Central or at a command prompt.

   - To start Interactive SQL from SQL Central, right-click the **`MLconsolidated - DBA`** database and click *Open Interactive SQL*.
   - To start Interactive SQL at a command prompt, run the following command:

     ```
     dbisql -c "DSN=mlxml_db"
     ```

2. Execute the following SQL statement in Interactive SQL to create the RemoteOrders table:

   ```
   CREATE TABLE RemoteOrders (
       order_id           INTEGER NOT NULL,
       product_id         INTEGER NOT NULL,
       quantity           INTEGER,
       order_status       VARCHAR(10) DEFAULT 'new',
       last_modified      TIMESTAMP DEFAULT CURRENT TIMESTAMP,
       PRIMARY KEY(order_id)
   );
   ```

   Interactive SQL creates the RemoteOrders table in your consolidated database.

3. Execute the following statement in Interactive SQL to create MobiLink system tables and stored procedures.

   Replace *C:\Program Files\SQL Anywhere 17*\ with the location of your SQL Anywhere 17 installation.

   ```
   READ "C:\Program Files\SQL Anywhere 17\MobiLink\setup\syncsa.sql";
   ```

   Interactive SQL applies `syncsa.sql` to your consolidated database. Running `syncsa.sql` creates a series of system tables and stored procedures prefaced with **ml_**. The MobiLink server works with these tables and stored procedures in the synchronization process.

Results

The RemoteOrders table is created and the MobiLink system tables and stored procedures are installed.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Synchronizing with XML [page 247]

Previous task: Lesson 2: Setting up Your MobiLink Consolidated Database [page 249]

Next task: Lesson 4: Adding Synchronization Scripts [page 253]

Related Information

CREATE TABLE Statement

1.5.8.4 Lesson 4: Adding Synchronization Scripts

In this lesson, you add scripts to your consolidated database for SQL row handling and direct row handling.

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 row handling allows you to synchronize remote data with tables in your MobiLink consolidated database. SQL-based scripts define:

- How data that is uploaded from a MobiLink client is to be applied to the consolidated database.
- What data should be downloaded from the consolidated database.

In this lesson, you write synchronization scripts for the following SQL-based upload and download events:

upload_insert

This event defines how new orders inserted in a client database should be applied to the consolidated database.

download_cursor

This event defines the orders that should be downloaded to remote clients.

download_delete_cursor

This event is required when using synchronization scripts that are not upload-only. You set the MobiLink server to ignore this event for the purpose of this tutorial.

You use direct row handling to add special processing to a SQL-based synchronization system. In this lesson, you register method names corresponding to the handle_UploadData, download_cursor, and download_delete_cursor events.

Procedure

1. Connect to your consolidated database in Interactive SQL if you are not already connected.

   Run the following command:

   ```
   dbisql -c "DSN=mlxml_db"
   ```

2. Use the ml_add_table_script stored procedure to add SQL-based table scripts for the upload_insert, download_cursor and download_delete_cursor events.

   Execute the following SQL statement in Interactive SQL. The upload_insert script inserts the uploaded order_id, product_id, quantity, and order_status into the MobiLink consolidated database. The download_cursor script uses timestamp-based filtering to download updated rows to remote clients.

   ```
   CALL ml_add_table_script( 'default', 'RemoteOrders',
       'upload_insert',
       'INSERT INTO RemoteOrders( order_id, product_id, quantity, order_status)
       VALUES( {ml r.order_id}, {ml r.product_id}, {ml r.quantity}, {ml
   r.order_status} )' );

   CALL ml_add_table_script( 'default', 'RemoteOrders',
       'download_cursor',
       'SELECT order_id, product_id, quantity, order_status
       FROM RemoteOrders WHERE last_modified >= {ml s.last_table_download}');
   CALL ml_add_table_script( 'default', 'RemoteOrders',
       'download_delete_cursor', '--{ml_ignore}');
   COMMIT;
   ```

3. Register the Java method for the handle_UploadData event.

   Execute the following SQL statement in Interactive SQL:

   ```
   CALL ml_add_java_connection_script( 'default',
       'handle_UploadData', 'MobiLinkOrders.GetUpload' );
   ```

   Interactive SQL registers the GetUpload method for the handle_UploadData event. You create the GetUpload method, which retrieves inserted data from the OrderComments table in the MobiLink client database, in an upcoming lesson.

4. Register the download_cursor and download_delete_cursor events.

   Execute the following SQL statements in Interactive SQL:

   ```
   CALL ml_add_table_script( 'default', 'OrderComments',
       'download_cursor', '--{ml_ignore}');
   CALL ml_add_table_script( 'default', 'OrderComments',
       'download_delete_cursor', '--{ml_ignore}');
   ```

The download_cursor and download_delete_cursor events must be registered for the OrderComments table when using scripts because the synchronization is bi-directional and not upload-only.

5. Commit your changes.

   Execute the following SQL statement in Interactive SQL:

   ```
   COMMIT;
   ```
6. Close Interactive SQL.

Results

Method names corresponding to the handle_UploadData, handle_DownloadData, end_download, download_cursor, and download_delete_cursor events are registered.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Synchronizing with XML [page 247]

Previous task: Lesson 3: Creating a Table in Your MobiLink Consolidated Database [page 251]

Next task: Lesson 5: Creating a Java Class for MobiLink Direct Row Handling [page 256]

Related Information

Overview of MobiLink Events
Script Additions and Deletions
Scripts to Upload Rows
Scripts to Download Rows
Scripts Required for Synchronization
Direct Row Handling
Direct Uploads
Direct Downloads
Partitioned Rows Among Remote Databases
Implementing Timestamp-based Downloads
upload_insert Table Event
upload_update Table Event
upload_delete Table Event
download_cursor Table Event
download_delete_cursor Table Event

1.5.8.5 Lesson 5: Creating a Java Class for MobiLink Direct Row Handling

In this lesson, you use direct row handling to process rows in the OrderComments table in your client database. You add the GetUpload methods for direct row handling for the handle_UploadData event. GetUpload writes uploaded comments to the XML file.

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

The following procedure shows you how to create a Java class including your methods for processing. For a complete listing, see the MobiLinkOrders Java code listing.

Procedure

1. Create a class named MobiLinkOrders.

   Write the following code:

   ```
   import com.sap.ml.script.*;
   import java.io.*;
   import java.sql.*;
   import javax.xml.parsers.DocumentBuilder;
   import javax.xml.parsers.DocumentBuilderFactory;
   import javax.xml.parsers.ParserConfigurationException;
   import org.xml.sax.SAXException;
   import org.w3c.dom.Document;
   import org.w3c.dom.Element;
   import org.w3c.dom.Node;
   import org.w3c.dom.NodeList;
   // For write operation
   import javax.xml.transform.Transformer;
   import javax.xml.transform.TransformerException;
   import javax.xml.transform.TransformerFactory;
   import javax.xml.transform.TransformerConfigurationException;
   import javax.xml.transform.dom.DOMSource;
   import javax.xml.transform.stream.StreamResult;
   public class MobiLinkOrders {
   ```
2. Declare a class-level DBConnectionContext instance and Document instance. Document is a class that represents an XML document as an object.

Write the following code:

```
     // Class level DBConnectionContext
     DBConnectionContext _cc;
     Document _doc;
```

The MobiLink server passes a DBConnectionContext instance to your class constructor. DBConnectionContext encapsulates information about the current connection with the MobiLink consolidated database.

3. Create your class constructor.

   Your class constructor sets your class-level DBConnectionContext instance.

   Write the following code:

```
     public MobiLinkOrders( DBConnectionContext cc ) throws IOException,
 FileNotFoundException {
         // Declare a class-level DBConnectionContext
         _cc = cc;
     }
```

4. Write the GetUpload method.

   The GetUpload method obtains an UploadedTableData class instance representing the OrderComments table. The OrderComments table contains special comments made by remote sales employees. You create this table in a later lesson.

   The UploadedTableData getInserts method returns a result set for new order comments.

   a. Write the method declaration.

      Write the following code:

```
     //  Method for the handle_UploadData synchronization event
     public void GetUpload( UploadData ut ) throws SQLException,
 IOException {
```

   b. Write code that retrieves any uploaded inserts from the MobiLink client.

      Write the following code:

```
         // Get an UploadedTableData for the remote table
         UploadedTableData remoteOrdersTable =
 ut.getUploadedTableByName("OrderComments");
         // Get inserts uploaded by the MobiLink client
         // as a java.sql.ResultSet
         ResultSet insertResultSet = remoteOrdersTable.getInserts();
```

   c. Write code that reads the existing XML file, `order_comments.xml`.

      Write the following code:

```
         try {
         readDom("order_comments.xml");
```

   d. Write code that adds all uploaded inserts to the XML file.

      Write the following code:

```
    // Write out each insert in the XML file
    while( insertResultSet.next() ) {
       buildXML(insertResultSet);
    }
```

e. Write code that outputs to the XML file.

   Write the following code:

```
            writeXML();
        }
```

f. Write code that closes the ResultSet.

   Write the following code:

```
            finally {
        // Close the result set of uploaded inserts
        insertResultSet.close();
    }
    }
```

5. Write the buildXML method.

   Write the following code:

```
      private void buildXML( ResultSet rs ) throws SQLException {
         int order_id = rs.getInt(1);
         int comment_id = rs.getInt(2);
         String order_comment = rs.getString(3);
         // Create the comment object to be added to the XML file
         Element comment = _doc.createElement("comment");
         comment.setAttribute("id", Integer.toString(comment_id));
         comment.appendChild(_doc.createTextNode(order_comment));
         // Get the root element (orders)
         Element root = _doc.getDocumentElement();

         // Get each individual order
         NodeList rootChildren = root.getChildNodes();

         for(int i = 0; i < rootChildren.getLength(); i++) {
             // If the order exists, add the comment to the order
             Node n = rootChildren.item(i);
             if(n.getNodeType() == Node.ELEMENT_NODE) {
                 Element e = (Element) n;
                 int idIntVal = Integer.parseInt(e.getAttribute("id"));

                 if(idIntVal == order_id) {
                     e.appendChild(comment);
                     // The comment has been added to the file, so exit
                     // the function.
                     return;
                 }
             }
         }
         // If the order did not exist already, create it
         Element order = _doc.createElement("order");
         order.setAttribute("id", Integer.toString(order_id));
         // Add the comment to the new order
         order.appendChild(comment);
         root.appendChild(order);
     }
```

6. Write the writeXML method.

   Write the following code:

```
     private void writeXML() {
         try {
             // Use a Transformer for output
             TransformerFactory tFactory = TransformerFactory.newInstance();
             Transformer transformer = tFactory.newTransformer();

             // The XML source is _doc
             DOMSource source = new DOMSource(_doc);
             // Write the xml data to order_comments.xml
             StreamResult result = new StreamResult(new
 File("order_comments.xml"));
             transformer.transform(source, result);
         } catch (TransformerConfigurationException tce) {
             // Error generated by the parser
             System.out.println ("\n** Transformer Factory error");
             System.out.println("   " + tce.getMessage() );
             // Use the contained exception, if any
             Throwable x = tce;
             if (tce.getException() != null) x = tce.getException();
             x.printStackTrace();
         } catch (TransformerException te) {
             // Error generated by the parser
             System.out.println ("\n** Transformation error");
             System.out.println("   " + te.getMessage() );
             // Use the contained exception, if any
             Throwable x = te;
             if (te.getException() != null) x = te.getException();
             x.printStackTrace();
         }
     }
```

7. Write the readDom method.

   Write the following code:

```
     private void readDom(String filename) {
         DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
         try {
             //parse the Document data into _doc
             DocumentBuilder builder = factory.newDocumentBuilder();
             _doc = builder.parse( new File(filename) );

         } catch (SAXException sxe) {
             // Error generated during parsing)
             Exception x = sxe;
             if (sxe.getException() != null) x = sxe.getException();
             x.printStackTrace();
         } catch (ParserConfigurationException pce) {
             // Parser with specified options can't be built
             pce.printStackTrace();
         } catch (IOException ioe) {
             // I/O error
             ioe.printStackTrace();
         }
     }
 }
```

8. Save your Java code as `MobiLinkOrders.java` in your working directory, `c:\MLobjxml`.

   To verify the code in `MobiLinkOrders.java`, see the MobiLinkOrders Java code listing.

9. Compile your class file.

   a. Navigate to the directory containing your Java source files.

   b. Compile MobiLinkOrders that refer to the MobiLink server API library for Java.

      You need to reference `mlscript.jar` located in *%SQLANY17%*`\Java` and make sure that you have the XML DOM library installed correctly.

Run the following command, replacing *C:\Program Files\SQL Anywhere 17* with your SQL Anywhere 17 directory:

```
javac -classpath "C:\Program Files\SQL Anywhere 17\java\mlscript.jar"
MobiLinkOrders.java
```

Results

The Java class for MobiLink direct row handling is created.

Next Steps

Proceed to the next lesson.

In this section:

MobiLinkOrders Java Code Listing [page 260]
The following listing shows the complete Java MobiLinkOrders class code used for this tutorial.

Task overview: Tutorial: Synchronizing with XML [page 247]

Previous task: Lesson 4: Adding Synchronization Scripts [page 253]

Next task: Lesson 6: Starting the MobiLink Server [page 263]

Related Information

Direct Row Handling
Synchronization Script Writing in Java

1.5.8.5.1 MobiLinkOrders Java Code Listing

The following listing shows the complete Java MobiLinkOrders class code used for this tutorial.

```
import com.sap.ml.script.*;
    import java.io.*;
    import java.sql.*;

    import javax.xml.parsers.DocumentBuilder;
    import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.xml.sax.SAXException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
// For write operation
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
public class MobiLinkOrders {
    // Class level DBConnectionContext
    DBConnectionContext _cc;
    Document _doc;

    public MobiLinkOrders( DBConnectionContext cc ) throws IOException,
FileNotFoundException {
        // Declare a class-level DBConnectionContext
        _cc = cc;
    }
    //  Method for the handle_UploadData synchronization event
    public void GetUpload( UploadData ut ) throws SQLException, IOException {
        // Get an UploadedTableData for the remote table
        UploadedTableData remoteOrdersTable =
ut.getUploadedTableByName("OrderComments");
        // Get inserts uploaded by the MobiLink client
        // as a java.sql.ResultSet
        ResultSet insertResultSet = remoteOrdersTable.getInserts();

    try {
        readDom("order_comments.xml");

        // Write out each insert in the XML file
        while( insertResultSet.next() ) {
        buildXML(insertResultSet);
        }
        writeXML();
    } finally {
        // Close the result set of uploaded inserts
        insertResultSet.close();
    }
    }

    private void buildXML( ResultSet rs ) throws SQLException {
        int order_id = rs.getInt(1);
        int comment_id = rs.getInt(2);
        String order_comment = rs.getString(3);
        // Create the comment object to be added to the XML file
        Element comment = _doc.createElement("comment");
        comment.setAttribute("id", Integer.toString(comment_id));
        comment.appendChild(_doc.createTextNode(order_comment));
        // Get the root element (orders)
        Element root = _doc.getDocumentElement();

        // Get each individual order
        NodeList rootChildren = root.getChildNodes();

        for(int i = 0; i < rootChildren.getLength(); i++) {
            // If the order exists, add the comment to the order
            Node n = rootChildren.item(i);
            if(n.getNodeType() == Node.ELEMENT_NODE) {
                Element e = (Element) n;
                int idIntVal = Integer.parseInt(e.getAttribute("id"));

                if(idIntVal == order_id) {
                    e.appendChild(comment);
                    // The comment has been added to the file, so exit
                    // the function
                    return;
                }
            }
        }
        // If the order did not exist already, create it
        Element order = _doc.createElement("order");
        order.setAttribute("id", Integer.toString(order_id));
        // Add the comment to the new order
        order.appendChild(comment);
        root.appendChild(order);
    }
    private void writeXML() {
        try {
            // Use a Transformer for output
            TransformerFactory tFactory = TransformerFactory.newInstance();
            Transformer transformer = tFactory.newTransformer();

            // The XML source is _doc
            DOMSource source = new DOMSource(_doc);
            // Write the xml data to order_comments.xml
            StreamResult result = new StreamResult(new
File("order_comments.xml"));
            transformer.transform(source, result);
        } catch (TransformerConfigurationException tce) {
            // Error generated by the parser
            System.out.println ("\n** Transformer Factory error");
            System.out.println("   " + tce.getMessage() );
            // Use the contained exception, if any
            Throwable x = tce;
            if (tce.getException() != null) x = tce.getException();
            x.printStackTrace();
        } catch (TransformerException te) {
            // Error generated by the parser
            System.out.println ("\n** Transformation error");
            System.out.println("   " + te.getMessage() );
            // Use the contained exception, if any
            Throwable x = te;
            if (te.getException() != null) x = te.getException();
            x.printStackTrace();
        }
    }
    private void readDom(String filename) {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        try {
            //parse the Document data into _doc
            DocumentBuilder builder = factory.newDocumentBuilder();
            _doc = builder.parse( new File(filename) );

        } catch (SAXException sxe) {
            // Error generated during parsing)
            Exception x = sxe;
            if (sxe.getException() != null) x = sxe.getException();
            x.printStackTrace();
        } catch (ParserConfigurationException pce) {
            // Parser with specified options can't be built
            pce.printStackTrace();
        } catch (IOException ioe) {
            // I/O error
            ioe.printStackTrace();
        }
    }
}
```

1.5.8.6 Lesson 6: Starting the MobiLink Server

In this lesson, you start the MobiLink server. You start the MobiLink server (mlsrv17) using the -c option to connect to your consolidated database, and the -sl java option to load your Java class.

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

Connect to your consolidated database and load the class on the mlsrv17 command line.

Replace `c:\MLobjxml` with the location of your source files and then run the following command:

```
mlsrv17 -c "DSN=mlxml_db" -o serverOut.txt -v+ -dl -zu+ -x tcpip  -sl java (-cp
c:\MLobjxml)
```

Below is a description of each MobiLink server option used in this tutorial. The options -o, -v, and -dl provide debugging and troubleshooting information. Using these logging options is appropriate in a development environment. For performance reasons, -v+ and -dl are typically not used in production.

Option	Description
-c	Precedes the connection string.
-o	Specifies the message log file `serverOut.txt`.
-v+	The -v option specifies what information is logged. Using -v+ sets maximum verbose logging.
-dl	Displays all log messages on screen.
-zu+	Adds new users automatically.
-x	Sets the communications protocol and parameters for MobiLink clients.
-sl java	Specifies a set of directories to search for class files, and forces the Java VM to load on server startup.

The MobiLink server messages window appears.

Results

The MobiLink server is started.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Synchronizing with XML [page 247]

Previous task: Lesson 5: Creating a Java Class for MobiLink Direct Row Handling [page 256]

Next task: Lesson 7: Setting up Your MobiLink Client Database [page 264]

Related Information

MobiLink Server Options
-sl java mlsrv17 Option

1.5.8.7 Lesson 7: Setting up Your MobiLink Client Database

In this lesson, you use a SQL Anywhere database for your consolidated database and your MobiLink client. For tutorial purposes your MobiLink client, consolidated database, and MobiLink server all reside on the same computer.

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

To set up the MobiLink client database, you create the RemoteOrders and OrderComments tables. The RemoteOrders table corresponds to the RemoteOrders table on the consolidated database. The MobiLink server uses SQL-based scripts to synchronize remote orders. The OrderComments table is only used on client databases. The MobiLink server processes the OrderComments tables using special events.

Procedure

1. Create your MobiLink client database using the dbinit command line utility.

   Navigate to `c:\MLobjxml` and then run the following command:

   ```
   dbinit -i -k -dba DBA,passwd remote1
   ```

   The -i and -k options omit jConnect support and Watcom SQL compatibility views, respectively.
2. Start your MobiLink client database using the dbsrv17 command line utility.

   Run the following command:

   ```
   dbsrv17 remote1
   ```
3. Connect to your MobiLink client database using Interactive SQL.

   Run the following command:

   ```
   dbisql -c "SERVER=remote1;UID=DBA;PWD=passwd"
   ```
4. Create the RemoteOrders table.

   Execute the following SQL statement in Interactive SQL:

   ```
   CREATE TABLE RemoteOrders (
       order_id            INTEGER NOT NULL,
       product_id          INTEGER NOT NULL,
       quantity            INTEGER,
       order_status        VARCHAR(10) DEFAULT 'new',
       PRIMARY KEY(order_id)
   );
   ```
5. Create the OrderComments table.

   Execute the following SQL statement in Interactive SQL:

   ```
   CREATE TABLE OrderComments (
       comment_id          INTEGER NOT NULL,
       order_id            INTEGER NOT NULL,
       order_comment       VARCHAR(255),
       PRIMARY KEY(comment_id),
       FOREIGN KEY(order_id) REFERENCES RemoteOrders(order_id)
   );
   ```
6. Create your MobiLink synchronization user, publication, and subscription.

   Execute the following SQL statement in Interactive SQL:

   ```
   CREATE SYNCHRONIZATION USER ml_sales1;
   CREATE PUBLICATION order_publ (TABLE RemoteOrders, TABLE OrderComments);
   CREATE SYNCHRONIZATION SUBSCRIPTION TO order_publ FOR ml_sales1
    TYPE TCPIP ADDRESS 'host=localhost';
   ```

   i Note

   You specify how to connect to the MobiLink server using the TYPE and ADDRESS clauses in the CREATE SYNCHRONIZATION SUBSCRIPTION statement.

You can use publications to determine what data is synchronized. In this case you specify the entire RemoteOrders and OrderComments tables.

Results

The remote database is created and set up for synchronization.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Synchronizing with XML [page 247]

Previous task: Lesson 6: Starting the MobiLink Server [page 263]

Next task: Lesson 8: Synchronizing [page 266]

Related Information

MobiLink Clients
Initialization Utility (dbinit)
CREATE SYNCHRONIZATION USER Statement [MobiLink]
CREATE PUBLICATION Statement [MobiLink] [SQL Remote]
CREATE SYNCHRONIZATION SUBSCRIPTION Statement [MobiLink]

1.5.8.8 Lesson 8: Synchronizing

In this lesson you use the dbmlsync utility to initiate MobiLink synchronization. Before starting dbmlsync, add order data and comments to your remote 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. Connect to the MobiLink client database from Interactive SQL if you are not already connected.

   Run the following command:

   ```
   dbisql -c "SERVER=remote1;UID=DBA;PWD=passwd"
   ```
2. Add an order to the RemoteOrders table in the client database.

   Execute the following SQL statement in Interactive SQL:

   ```
   INSERT INTO RemoteOrders (order_id, product_id, quantity, order_status)
    VALUES (1,12312,10,'new');
   ```
3. Add a comment to the OrderComments table in the client database.

   Execute the following SQL statement in Interactive SQL:

   ```
   INSERT INTO OrderComments (comment_id, order_id, order_comment)
    VALUES (1,1,'send promotional material with the order');
   ```
4. Commit your changes.

   Execute the following SQL statement in Interactive SQL:

   ```
   COMMIT;
   ```
5. Run the following command at a command prompt:

   ```
   dbmlsync -c "SERVER=remote1;UID=DBA;PWD=passwd" -o rem1.txt -v+
   ```

   The following table contains a description for each dbmlsync option used in this lesson.

Option	Description
-c	Specifies the connection string.
-o	Specifies the message log file `rem1.txt`.
-v+	The -v option specifies what information is logged. Using -v+ sets maximum verbose logging.

   Once you have started the MobiLink synchronization client, an output screen appears indicating that the synchronization succeeded.
6. SQL-based synchronization transferred rows in the client RemoteOrders table to the RemoteOrders table in the consolidated database.

   Perform the following steps to verify that the information added to the client RemoteOrders table was transferred to the RemoteOrders table in the consolidated database:

   a. To start Interactive SQL at a command prompt, run the following command:

   ```
   dbisql -c "DSN=mlxml_db"
   ```

   b. Execute the following SQL statement in Interactive SQL:

   ```
   SELECT * FROM RemoteOrders;
   ```

7. Java processing inserted your comment in the XML file.

   Go to `c:\MLobjxml` and open `order_comments.xml` in a text editor to verify that the comment was inserted.

Results

The remote and consolidated databases are synchronized.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Synchronizing with XML [page 247]

Previous task: Lesson 7: Setting up Your MobiLink Client Database [page 264]

Next task: Lesson 9: Cleaning Up [page 268]

Related Information

SQL Anywhere Clients
MobiLink SQL Anywhere Client Utility (dbmlsync) Syntax

1.5.8.9 Lesson 9: Cleaning Up

Remove the tutorials materials from your computer.

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. Close all instances of Interactive SQL.
2. Close the SQL Anywhere, MobiLink, and synchronization client windows.
3. Delete all tutorial-related ODBC data sources.
    a. Start the ODBC Administrator.

       Run the following command:

       ```
       odbcad32
       ```
    b. Remove the **`mlxml_db`** data source.
4. Delete the consolidated and remote databases.
    a. Navigate to the directory containing your consolidated and remote databases.
    b. Delete `MLconsolidated.db`, `MLconsolidated.log`, `remote1.db`, and `remote1.log`.

Results

The tutorial materials are removed from your computer.

Task overview: Tutorial: Synchronizing with XML [page 247]

Previous task: Lesson 8: Synchronizing [page 266]

1.5.9 Tutorial: Using Central Administration of Remote Databases

This tutorial leads you through the process of setting up central administration of remote databases and demonstrates how several common operations can be performed.

Prerequisites

This tutorial assumes you have a complete install of SQL Anywhere, including MobiLink and SQL Central on your local computer where you are running the tutorial.

You must have the following roles and privileges on the consolidated database:

- SYS_AUTH_RESOURCE_ROLE compatibility role
- MONITOR system privilege

You must have the following roles and privileges on the remote database:

- SYS_REPLICATION_ADMIN_ROLE system role
- SYS_RUN_REPLICATION_ROLE system role

Context

You may follow this tutorial to either set up central administration from scratch or to add central administration to an existing synchronization system. Throughout the procedure, the tutorial points out where you should do different things if you are adding central administration to an existing synchronization system.

This tutorial shows you how to:

- Create a consolidated database and MobiLink project.
- Start the MobiLink server, define a MobiLink user and an Agent, and configure the Agent on the remote device.
- Create and deploy a synchronization model.
- Work with remote tasks.

Several introductory and tutorial videos on central administration of remote databases are available online

> **i Note**
>
> The video tutorials are based on version 12.0.0 of SQL Anywhere. Some visuals and procedures may differ from SQL Anywhere 17.0.11.

1. Lesson 1: Creating a Consolidated Database [page 272]
   In this lesson, you set up a consolidated database. If you have an existing synchronization system, proceed to Lesson 2: Creating a MobiLink project.
2. Lesson 2: Creating a MobiLink Project [page 273]
   To perform central administration, you must create a MobiLink project. The project acts as a container for the various objects you define for central administration.
3. Lesson 3: Starting the MobiLink Server [page 274]
   In this lesson, you start the MobiLink server. The MobiLink server is needed both to synchronize data from your remote database and to synchronize tasks and task results between the consolidated database and the agent database on each remote device.
4. Lesson 4: Defining a MobiLink User [page 276]
   In this lesson, you define a MobiLink user for the Agent to use. You can skip this lesson if you have an existing synchronization system, and you want the MobiLink Agent to use one of your existing MobiLink users to synchronize.
5. Lesson 5: Defining an Agent [page 277]
   In this lesson, you define an Agent. This Agent represents an instance of the MobiLink Agent running on a remote device.
6. Lesson 6: Configuring the Agent on the Remote Device [page 279]
   In this lesson, you run the MobiLink Agent. The MobiLink Agent must be running on each remote device that is centrally administered. For this tutorial, the Agent runs on the same computer where the MobiLink server is running.

7. Lesson 7: Creating a Synchronization Model [page 281]
    In this lesson, you create a synchronization model.
8. Lesson 8: Deploying the Synchronization Model [page 282]
    In this lesson, you deploy the synchronization model you created in the previous lesson.
9. Lesson 9: Creating a Remote Task [page 284]
    In this lesson, you create a remote task to display the message "Hello World" on the remote device.
10. Lesson 10: Deploying a Remote Task [page 285]
    In this lesson, you deploy the remote task so that it can be assigned to an Agent.
11. Lesson 11: Checking the Status of a Remote Task [page 286]
    In this lesson, you check the status of your remote task in SQL Central.
12. Lesson 12: Creating a Remote Database on a Remote Device [page 287]
    In this lesson, you use a remote task to create a new remote database on the remote device. If you are adding central administration to an existing synchronization system, proceed to the lesson on scheduling synchronization.
13. Lesson 13: Scheduling Synchronization [page 289]
    The next step is to configure the MobiLink Agent to synchronize its remote database at regular intervals. You do this by creating a remote task that executes based on a schedule and synchronizes the database each time it executes.
14. Lesson 14: Modifying Scheduled Synchronizations [page 291]
    In the last lesson you created a remote task to synchronize the remote database once per minute. In this lesson, you change the synchronization interval to once per hour.
15. Lesson 15: Forcing Immediate Synchronization [page 292]
    Previously, you set up the remote database to synchronize once per hour. This lesson shows you how to use a server-initiated remote task (SIRT) to force a synchronization before the hour is up. This technique is useful whenever you want to centrally control when a certain task executes.
16. Lesson 16: Changing the Remote Schema [page 293]
    In this lesson, you change the schema of the remote database. For the purposes of this tutorial, a schema change occurs whenever you change the remote schema name of the database. You are never forced to change the remote schema name, it is always left up to your discretion.
17. Lesson 17: Querying the Remote Database [page 295]
    In this lesson, you query the remote database and return results to the server. This is very useful when troubleshooting because you can find out exactly what state the remote database is in.
18. Lesson 18: Uploading Files Using SIRT [page 297]
    In this lesson, you upload files from the remote device using a server-initiated remote task (SIRT). Uploading files from the remote device is useful for troubleshooting because an administrator can examine the files for problems.
19. Lesson 19: Cleaning Up [page 299]
    Remove all tutorial materials from your computer.

Related Information

Manage Remote Databases
SQL Anywhere MobiLink Client Deployment
UltraLite MobiLink Client Deployment

1.5.9.1 Lesson 1: Creating a Consolidated Database

In this lesson, you set up a consolidated database. If you have an existing synchronization system, proceed to Lesson 2: Creating a MobiLink project.

Prerequisites

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

Procedure

1. Run the following commands to create directories to be used in this tutorial. The consolidated directory contains all the database and other files that would normally reside on your central server.

```
md c:\cadmin_demo
md c:\cadmin_demo\consolidated
```

2. Create a SQL Anywhere consolidated database and an ODBC data source to connect to it.

```
cd c:\cadmin_demo\consolidated
dbinit -dba DBA,passwd consol.db
start dbsrv17 consol.db
dbdsn -w cadmin_tutorial_consol consol -y -c
"UID=DBA;PWD=passwd;DBF=consol.db;SERVER=consol"
cd ..
```

3. Connect to the database in Interactive SQL. Run the following command:

```
dbisql -c "DSN=cadmin_tutorial_consol"
```

4. Execute the following statement in Interactive SQL to create MobiLink system tables and stored procedures using the `syncsa.sql` setup script. Replace `C:\Program Files\`*SQL Anywhere 17*`\` with the location of your SQL Anywhere 17 installation.

```
READ "C:\Program Files\SQL Anywhere 17\MobiLink\setup\syncsa.sql";
```

5. Close *Interactive SQL*. You do not need to save your SQL statements.

Results

A SQL Anywhere database is created and you are connected to it.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Next task: Lesson 2: Creating a MobiLink Project [page 273]

1.5.9.2 Lesson 2: Creating a MobiLink Project

To perform central administration, you must create a MobiLink project. The project acts as a container for the various objects you define for central administration.

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. To start SQL Central, click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Central*.
2. Click *Tools* > *MobiLink 17* > *New Project*.

   The *Create Project Wizard* appears.
3. On the *Welcome* page, change the project name to **`Central Admin Tutorial`** and type `C:/cadmin` for the location for the Project file. Click *Next*.
4. On the *Specify a Consolidated Database* page, type **`Tutorial`** for the *Database display name*.
5. Enter the following values for the *Connection string*:

   ```
   UID=DBA;PWD=passwd;DSN=cadmin_tutorial_consol
   ```
6. Choose *Remember the password* and click *Next*.

   You may receive a warning that a synchronization model will not be created because the database does not contain any tables. Click *OK*.
7. Select *Add a remote schema name to the project* and type **`Tutorial Application v1.0`** for the schema name. Click *Next*.
8. Select *SQL Anywhere* for the database type and click *Finish*.

If this is the first time the consolidated database has been used by MobiLink, a message appears asking you to install the MobiLink system setup. Installing the MobiLink system setup adds MobiLink system tables and procedures. Click *Yes*, and then click *OK*.

Results

The MobiLink project is created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 1: Creating a Consolidated Database [page 272]

Next task: Lesson 3: Starting the MobiLink Server [page 274]

1.5.9.3 Lesson 3: Starting the MobiLink Server

In this lesson, you start the MobiLink server. The MobiLink server is needed both to synchronize data from your remote database and to synchronize tasks and task results between the consolidated database and the agent database on each remote device.

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

If you have an existing synchronization system you can skip this lesson since you already have the server running. However, check your server command line and ensure that the -ftr and -ftru options are specified. These options are required to download files to your remote devices and to upload files from your remote devices.

Procedure

At a command prompt, run the following commands:

```
md c:\cadmin_demo\consolidated\upload
md c:\cadmin_demo\consolidated\download
cd c:\cadmin_demo\consolidated
start mlsrv17.exe -c "DSN=cadmin_tutorial_consol;UID=DBA;PWD=passwd" -ftr
download -ftru upload -x tcpip(port=2439) -v+ -ot mlsrv.txt
cd ..
```

Following is a summary of the options used:

-c

Specifies the connection parameters MobiLink uses to connect to the consolidated database.

-ftr

Specifies the directory where MobiLink looks for files to download.

-ftru

Specifies the directory where MobiLink puts files that are uploaded.

-x

Specifies communication parameters that define how synchronization clients may connect to the MobiLink server.

-v+

Specifies maximum verbosity. This setting is helpful for debugging but can slow performance in a production environment.

-ot

Specifies the file where MobiLink output messages are logged.

Results

The MobiLink server is started and the upload and download directories that contain files to be uploaded from or downloaded to remote devices are created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 2: Creating a MobiLink Project [page 273]

Next task: Lesson 4: Defining a MobiLink User [page 276]

1.5.9.4 Lesson 4: Defining a MobiLink User

In this lesson, you define a MobiLink user for the Agent to use. You can skip this lesson if you have an existing synchronization system, and you want the MobiLink Agent to use one of your existing MobiLink users to synchronize.

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

When an Agent synchronizes its agent database, it must authenticate itself to the MobiLink server. It authenticates itself by using a MobiLink user and optionally a password. Normally you would use the same MobiLink user and password to synchronize your remote databases that the Agent uses to synchronize the agent database.

Procedure

1. In SQL Central, click *View* > *Folders*.
2. Under MobiLink 17 expand *Central Admin Tutorial*, *Consolidated Databases*, **`Tutorial`**.
3. Right-click *Users* and click *New* > *User*.

   The *Create User Wizard* appears.
4. On the *Welcome* page, type **`JOHN`** for the name of the new user and click *Next*
5. On the *Authentication* page, check *This user will require a password to connect when using standard MobiLink authentication* and type **`passwd`** in both the *Password* and *Confirm password* fields. Click *Finish*.

   If you do not want to authenticate Agents that try to synchronize, skip this step and add the -zu+ option to the MobiLink server command line. When -zu+ is specified, each MobiLink user is registered when it first attempts to synchronize.

Results

The MobiLink user is created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 3: Starting the MobiLink Server [page 274]

Next task: Lesson 5: Defining an Agent [page 277]

Related Information

-zu mlsrv17 Option

1.5.9.5 Lesson 5: Defining an Agent

In this lesson, you define an Agent. This Agent represents an instance of the MobiLink Agent running on a remote device.

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

You must create a separate Agent for each remote device you are managing.

Procedure

1. Under MobiLink 17 expand *Central Admin Tutorial*, *Consolidated Databases*, **`Tutorial`**.
2. Right-click *Agents* and click *New* > *Agent*.

   The *Create MobiLink Agent Wizard* appears.
3. On the *Welcome* page, choose *Set up a single agent* and click *Next*.

4. On the *Agent ID* page, type **`AID_JOHN`** for the *Agent ID*. The Agent ID can be any value you like, but each Agent must have a unique ID. By convention, Agent IDs begin with the prefix AID_ and usually the second part of the agent ID is the MobiLink user name used by the Agent. Optionally, you could enter a description for the Agent in the *Description* field. Click *Next*.
5. The *Remote Database* page lets you define a remote database to be managed by this Agent. This does not actually create the database; you do that later. For *Remote Schema Name*, select **`Tutorial Application v1.0`**, which is the name you defined in the previous lesson, from the dropdown list.
6. Enter the following connection string in the *Database Connection String* field:

   ```
   start=dbsrv17;SERVER=tutorial_v1;DBF={db_location}
   \tutorial_v1.db;UID=DBA;PWD=passwd
   ```

   This string value uses the macro {db_location}. This macro is replaced by the directory on the remote device where application databases are stored. Click *Next*.
7. On the *Agent Configuration* page, type **`30`** and choose *Seconds* for the *Synchronization interval*. The synchronization interval controls how frequently the Agent synchronizes its agent database. Synchronizing the agent database is how an Agent receives new tasks to perform and uploads the results of tasks it has already performed.
8. On the *Agent Configuration* page, type **`10`** and choose *Seconds* for the *Administration polling interval*. The administration polling interval determines how frequently the Agent checks for requests from the server for it to synchronize or perform other actions.

   > **i Note**
   >
   > The short values chosen for the synchronization interval and administration polling interval provide a very responsive Agent, which is important for a demonstration or for troubleshooting. However, using short values globally in a production system results in increased load on your server and reduced performance.

9. Click *Finish*.

Results

The Agent is created and configured.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 4: Defining a MobiLink User [page 276]

Next task: Lesson 6: Configuring the Agent on the Remote Device [page 279]

1.5.9.6 Lesson 6: Configuring the Agent on the Remote Device

In this lesson, you run the MobiLink Agent. The MobiLink Agent must be running on each remote device that is centrally administered. For this tutorial, the Agent runs on the same computer where the MobiLink server is running.

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. Create a directory that contains the files that would normally be on the remote device.

```
md c:\cadmin_demo\remote
cd c:\cadmin_demo\remote
```

2. Run the MobiLink Agent in configuration mode as follows:

```
mlagent -cr -db . -x tcpip{host=localhost;port=2439} -a AID_JOHN  -u JOHN -p
passwd
```

This step creates an agent database and stores some configuration information in it. Once the specified options are stored in the database, the Agent shuts down. Following is a summary of the options you used:

-cr

Specifies that the Agent should run in configuration mode and that it should discard any settings stored during previous runs in configuration mode.

-db

Specifies where the Agent should create application databases. This becomes the value of the {db_location} macro.

-x

Specifies how the Agent should connect to the MobiLink server to synchronize its agent database (to receive new tasks and upload results of tasks it has run). If you are adding central administration to an existing synchronization system, you need to change the value specified for this option to an appropriate string for connecting to your MobiLink server.

-a

Specifies the Agent ID for this Agent. You specified the same Agent ID that you previously created in the consolidated database using SQL Central.

-u

Specifies the MobiLink user the Agent uses when synchronizing the agent database. This value is used by the MobiLink server primarily to authenticate the Agent.

-p

Specifies the password that goes with the MobiLink user specified with the -u option.

3. Run the MobiLink Agent on the remote device. For this tutorial, you explicitly start the Agent running as follows:

```
start mlagent -v9 -ot agent.txt
```

Following is a summary of the options used to run the Agent in this lesson.

-v9

Uses maximum verbosity. Using this logging option is appropriate in a development environment. For performance reasons, -v9 is typically not used in a production environment.

-ot

Specifies the file where the Agent logs its output.

4. You should now have the MobiLink Agent running and it should be synchronizing successfully. To check, return to SQL Central. In the *Folders* view, under *MobiLink 17* expand *Central Admin Tutorial*. Expand *Consolidated Databases* and then choose **`Tutorial`**. Choose *Agents*, select *AID_JOHN* and look at the *Events* tab in the right pane. You should see an entry that indicates the Agent's first synchronization.

> i Note
>
> **Production considerations for Agent configuration**
>
> Keep the following considerations in mind when using central administration in a production environment:
>
> - You may need to change the values specified for the -u and -p options to an appropriate MobiLink user and password combination for your synchronization system.
> - You might want to use the -on option to limit the size of the Agent log file.
> - A remote device can only be remotely administered while the MobiLink Agent is running on it. You would likely want to take steps to ensure that the Agent is always running. Some strategies for this might include running the Agent as a service or adding the agent to the Run startup group in the registry.

Results

The MobiLink Agent is running and synchronizing.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 5: Defining an Agent [page 277]

Next task: Lesson 7: Creating a Synchronization Model [page 281]

1.5.9.7 Lesson 7: Creating a Synchronization Model

In this lesson, you create a synchronization model.

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

If you are adding central administration to existing synchronization system, proceed to Lesson 9: Deploying the synchronization model.

Procedure

1. Define the tables for the remote database in the consolidated database. In the *Folders* view of SQL Central under *MobiLink 17*, expand *Central Admin Tutorial*, *Consolidated Databases*. Right-click **`Tutorial - DBA`** and click *Open Interactive SQL*.
2. In the *SQL Statements* pane type the following:

```
CREATE TABLE customer(
    cust_id        INTEGER PRIMARY KEY,
    f_name         VARCHAR(100),
    l_name         VARCHAR(100)
)
```

3. Press F5 to execute the SQL. Close Interactive SQL. You do not need to save your SQL statements.
4. In the *Folders* view of SQL Central, right-click *Central Admin Tutorial* > *New* > *Synchronization Model*.
5. On the *Welcome* page, type **`tutorial1`** for the name of the new synchronization model. Click *Next*.
6. On the *Primary Key Requirements* page check all three checkboxes to confirm that your schema meets the requirements for synchronization. Click *Next*.
7. On the *Consolidated Database Schema* page, choose the **`Tutorial`** database and click *Next*.

8. On the *Remote Database Schema* page, select *No, create a new remote database schema* and click *Next*.
9. On the *New Remote Database Schema* page, ensure the customer table is selected and click *Next*. Click *Finish*.

Results

You have now created a synchronization model that contains a single table called `customer` that can be synchronized between the remote and the consolidated databases. The next step is to deploy that model to create synchronization objects in the consolidated database and to generate SQL for creating a remote database.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 6: Configuring the Agent on the Remote Device [page 279]

Next task: Lesson 8: Deploying the Synchronization Model [page 282]

1.5.9.8 Lesson 8: Deploying the Synchronization Model

In this lesson, you deploy the synchronization model you created in the previous lesson.

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 *Folders* view of SQL Central under *MobiLink 17*, expand *Central Admin Tutorial* > *Synchronization Models*. Right-click `tutorial1` and click *Deploy*.

   The *Deploy Synchronization Model Wizard* appears.

2. On the *Welcome* page, accept the default location to contain the files generated by the wizard and click *Next*.
3. On the *Client Network Options* page, choose the following options and then click *Next*.

   Protocol

   TCP/IP

   Host

   localhost

   Port

   `2439`
4. On the *MobiLink User And Password* page, select *Use macro values appropriate for remote tasks*:

   The *{ml_username}* and *{ml_password}* macro values are used in the generated SQL files and are replaced with the MobiLink user and password being used by the MobiLink Agent when the SQL is executed on the remote device. A synchronization profile is automatically created with the name tutorial1_{ml_username}, where the {ml_username} macro is replaced with the name of the MobiLink user, which in this case is JOHN.
5. On the *Synchronization Profile* page, type *tutorial1_JOHN* in the *Synchronization profile name* field.
6. Click *Next* until you get to the *Choose How To Prepare Databases For Synchronization* page and perform the following tasks:
   - For *What do you want to do with the SQL script created to prepare the consolidated database for synchronization*, select *Execute against consolidated database*.
   - For *What do you want to do with the SQL script created to prepare the remote database for synchronization*, select *Do not execute*.
   - Click *Next* and then click *Finish*.

   When you navigate away from the synchronization model, you are asked save your changes. Click *Yes*.

Results

You have now completed creating and deploying a synchronization model. When you deployed the model, scripts were added to the consolidated database to allow a remote database to synchronize. You also generated a SQL file in the `c:\cadmin_demo\Central Admin Tutorial\tutorial1_deploy\` directory, which can be used to create a remote database. You may like to look at those files now.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 7: Creating a Synchronization Model [page 281]

Next task: Lesson 9: Creating a Remote Task [page 284]

1.5.9.9 Lesson 9: Creating a Remote Task

In this lesson, you create a remote task to display the message "Hello World" on the remote device.

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

Most actions in central administration involve a remote task. A remote task is a collection of commands that is created by an administrator. It can be assigned to one or more Agents. Once assigned to an Agent, the remote task is downloaded to the Agent the next time the Agent synchronizes its agent database. The Agent then executes the task at an appropriate time and uploads information about the execution.

Procedure

1. Create a new remote task. In the *Folders* view of SQL Central under *MobiLink 17*, expand *Central Admin Tutorial*, right-click *Remote Tasks* and click *New* > *Remote Task*.

   The *Create Remote Task Wizard* appears.
2. On the *Welcome* page, type **`Hello World`** in the *Name* field. Click *Next*.
3. On the *Trigger Mechanisms* page, check *When it is received by an agent* and click *Finish* to complete the wizard.
4. Click the newly created **`Hello World`** task in the *Folders* view. In the right pane you see the *Commands* tab which allows you to add commands to your task.
5. On the *Commands* tab, choose *Prompt* from the *Command Type* dropdown list. In the *Message* field, type **`Hello World`**.
6. To add a second command to the task, either press Tab twice until a new command appears, or click the *Add command* button. Set the command type for the second command to *Prompt* and type **`Hello Again`** in the *Message* field.

   The Hello World task you just created is a design-time task. It is stored in the project on your local computer. Before you can assign the task to an Agent, you must copy it into the consolidated database by deploying it.

Results

The remote task is created and ready to be deployed.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 8: Deploying the Synchronization Model [page 282]

Next task: Lesson 10: Deploying a Remote Task [page 285]

1.5.9.10 Lesson 10: Deploying a Remote Task

In this lesson, you deploy the remote task so that it can be assigned to an Agent.

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 *Folders* view of SQL Central under *MobiLink 17*, expand *Central Admin Tutorial*, *Remote Tasks* and then right-click the *Hello World* task and click *Deploy*.

   The *Deploy Remote Task Wizard* is displayed.
2. Accept the defaults on the *Task Name And Destination* page and click *Next*.

   This gives the deployed task the same name as the design-time task, which is what you would normally want to do unless you are deploying the same design-time task for a second time. In that case, you would have to change the name for the deployed task.
3. The *Recipients* page lets you assign the deployed task to existing Agents. You can also do this later as a separate step. From the *Recipients* dropdown, select *Specific agents*. In the *Agent* list, select **`AID_JOHN`** and click *Next*.
4. On the *Delivery Options* page, check *The next time the agent synchronizes* and click *Next*.

5. On the *Reporting Results And Status* page, check *Send results and status immediately* for both questions. This ensures that you receive timely notification when your task executes. For routine tasks and repetitive tasks you may choose to receive feedback less quickly (especially on success), as this reduces the number of synchronizations of the agent database and the load on the MobiLink server.
6. Click *Finish*.

   The next time the Agent **`AID_JOHN`** synchronizes its agent database, it receives the new task and executes it. Click *OK* on the message boxes with the text *Hello World* and *Hello Again*.

   If you look at the *Folders* view, there are two copies of the **`Hello World`** task in the list. The deployed copy can be seen in the *Folders* view under *Remote Tasks* > *Deployed Tasks*. This is the copy in the consolidated database. The deployed copy of the task can no longer be changed. The design-time copy of the task is still visible under *Remote Tasks*. This task can be changed and can be deployed again with a new name.

   You can assign a deployed task to additional Agents at any time by right-clicking it and choosing *Add Recipients*.

Results

The remote task is executed.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 9: Creating a Remote Task [page 284]

Next task: Lesson 11: Checking the Status of a Remote Task [page 286]

1.5.9.11 Lesson 11: Checking the Status of a Remote Task

In this lesson, you check the status of your remote task in SQL Central.

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 *Folders* view of SQL Central under *MobiLink 17*, expand *Central Admin Tutorial* > *Remote Tasks* > *Deployed Tasks*. Click the deployed version of the `Hello World` task and select the *Results* tab in the right pane. Wait until the tab is automatically refreshed or press F5 to refresh immediately. On the *Results* tab there is a line for each command in the task, with a *Result Code* that indicates if the command succeeded or failed. A *Result Code* of 0 indicates success.
2. To see the results of a task execution displayed in different ways, select the *Recipients* tab for the deployed task, or look at the *Events* or *Tasks* tab of the Agent that executed the task.

Results

The results of the task execution are displayed.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 10: Deploying a Remote Task [page 285]

Next task: Lesson 12: Creating a Remote Database on a Remote Device [page 287]

1.5.9.12 Lesson 12: Creating a Remote Database on a Remote Device

In this lesson, you use a remote task to create a new remote database on the remote device. If you are adding central administration to an existing synchronization system, proceed to the lesson on scheduling synchronization.

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. Create a new remote task. In the *Folders* view of SQL Central under *MobiLink 17*, expand *Central Admin Tutorial*. Right-click *Remote Tasks* and click *New* > *Remote Task*.

   The *Create Remote Task Wizard* appears.
2. On the *Welcome* page, type **`Create DB`** in the *Name* field. Unlike the task you created before, this task creates or acts on a remote database, so check *This task requires or creates a remote database* and select the remote schema name **`Tutorial Application v1.0`**. This identifies the remote database that the database actions in this task act on. Click *Next*.
3. On the *Trigger Mechanisms* page, check *When it is received by an agent* then click *Finish* to complete the wizard.
4. Click the newly created **`Create DB`** task in the *Folders* view.
5. Add commands to the remote task from the *Commands* pane on the right.
   a. The first command creates a new, empty database on the remote device. Set the command type to *Create database*.
   b. Set the filename to **`{db_location}\tutorial_v1.db`**. This file name corresponds to the file name in the connection string you specified when you configured the Agent.
   c. Press Tab until a new command appears.
   d. The second command creates the schema in the new database. Set the command type to *Execute SQL*. Click *Import*.
   e. From the *Open* window, choose the file `c:\cadmin\Central Admin Tutorial\tutorial1_deploy\remote_setup.sql` and click *Open*. This imports the SQL for initializing a remote database that was generated when you deployed the synchronization model into the command.
6. The remote task is now complete. Deploy the task and assign it to the agent **`AID_JOHN`**:
   a. Right-click the **`Create DB`** task in the *Folders* view and click *Deploy*.

      The *Deploy Remote Task Wizard* appears.
   b. Accept the defaults on the *Task Name And Destination* page and click *Next*.
   c. From the *Recipients* dropdown, select *Specific agents*. In the *Agent* list, select **`AID_JOHN`** and click *Next*.
   d. On the *Delivery Options* page, check *The next time the agent synchronizes* and click *Next*.
   e. On the *Reporting Results And Status* page, select *Send results and status immediately* for both questions.
   f. Click *Finish*.
7. Check to see if the task was successful:
   a. In the *Folders* view of SQL Central under *MobiLink 17*, expand *Central Admin Tutorial* > *Consolidated Databases* > *Tutorial* > *Agents*, and click **`AID_JOHN`**.
   b. Select the *Events* tab and look for the **`Create DB`** task. Wait until the tab is automatically refreshed, or press F5 to refresh immediately.

Results

The new remote database is created on the remote device.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 11: Checking the Status of a Remote Task [page 286]

Next task: Lesson 13: Scheduling Synchronization [page 289]

1.5.9.13 Lesson 13: Scheduling Synchronization

The next step is to configure the MobiLink Agent to synchronize its remote database at regular intervals. You do this by creating a remote task that executes based on a schedule and synchronizes the database each time it executes.

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

This task is different from the other tasks you have created because the other tasks are executed only once. This tasks remains on the remote device and executes at regular intervals until you stop it.

Procedure

1. Create a new remote task. In the *Folders* view of SQL Central under *MobiLink 17*, expand *Central Admin Tutorial*. Right-click *Remote Tasks* and click *New* > *Remote Task*.

   The *Create Remote Task Wizard* appears.

2. On the *Welcome* page, type `Sync` in the *Name* field. Check *This task requires or creates a remote database* and select the remote schema name `Tutorial Application v1.0`. This identifies the remote database that the database actions in this task acts on. Click *Next*.
3. On the *Trigger Mechanisms* page, select *Based on a schedule* and click *Next*.
4. Accept the defaults on the *Start Time And Date* page. This allows the task to start running immediately. Click *Next*.
5. On the *Repetition* page, check *Repeat every* and set the interval to one minute. Click *Finish* to complete the wizard.
6. Click the newly created `Sync` task in the *Folders* view.
7. Add a single command to the task to cause a synchronization.
    a. On the *Commands* tab, set the *Command type* for the first command to *Synchronize*.
    b. For *Synchronization profile*, type `tutorial1_JOHN`. This is the synchronization profile that was created when you deployed the synchronization model.
8. The synchronization task is now complete. Right-click `Sync` and click *Deploy*. Click *Next*.
9. From the *Recipients* dropdown, click *Specific agents* and assign the task to agent `AID_JOHN`. Click *Next* and then click *Next* again.
10. On the *Reporting Results And Status* page, set *If task succeeds* to *Send only status later* and set *If task fails* to *Send results and status immediately*.

    Since this task repeats frequently, it is a good idea to limit the feedback requested to improve performance.
11. Click *Finish*.

Results

Once the Agent receives this new task, it begins to synchronize its remote database once each minute.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 12: Creating a Remote Database on a Remote Device [page 287]

Next task: Lesson 14: Modifying Scheduled Synchronizations [page 291]

1.5.9.14 Lesson 14: Modifying Scheduled Synchronizations

In the last lesson you created a remote task to synchronize the remote database once per minute. In this lesson, you change the synchronization interval to once per hour.

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

Once a remote task is deployed, the deployed version cannot be modified. Instead, you create a new remote task with the desired modifications, and then you cancel the existing task and deploy the new task to replace it.

Procedure

1. First, create a new remote task with the desired repeat interval using the existing deployed task as a template.
    a. In the *Folders* view of SQL Central under *MobiLink 17*, expand *Central Admin Tutorial* > *Remote Tasks* > *Deployed Tasks*. Right-click **Sync** and choose *Copy* to copy the task to the clipboard.
    b. Right-click *Remote Tasks* and choose *Paste*. A window appears asking for you to rename the remote task. Type **Sync every hour** and click *OK*.
    c. Right-click the new **Sync every hour** task and choose *Properties*. On the *Repetition* page of the properties window, change the *Repeat every* value from *1 minutes* to *1 hours* and click *OK*.
2. Next, cancel the existing remote task that causes synchronization each minute.
    a. In the *Folders* view, click the deployed version of the **Sync** task and click the *Recipients* tab in the right pane.
    b. Right-click the entry in the table for agent **AID_JOHN** and click *Cancel*.
3. Lastly, deploy the new **Sync every hour** task and assign it to agent **AID_JOHN**.
    a. Right-click **Sync every hour** and click *Deploy*. Click *Next*.
    b. From the *Recipients* dropdown list, click *Specific agents* and assign the task to agent **AID_JOHN**. Click *Next* and then click *Next* again.
    c. On the *Reporting Results and Status* page, set *If task succeeds* to *Send only status later* and set *If task fails* to *Send results and status immediately*.
    d. Click *Finish*.

Results

Once the Agent receives this new task, it begins to synchronize its remote database once an hour instead of once a minute.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 13: Scheduling Synchronization [page 289]

Next task: Lesson 15: Forcing Immediate Synchronization [page 292]

1.5.9.15 Lesson 15: Forcing Immediate Synchronization

Previously, you set up the remote database to synchronize once per hour. This lesson shows you how to use a server-initiated remote task (SIRT) to force a synchronization before the hour is up. This technique is useful whenever you want to centrally control when a certain task executes.

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

In the *Folders* view of SQL Central under *MobiLink 17*, expand *Central Admin Tutorial* > *Remote Tasks* > *Deployed Tasks*. Right-click `Sync every hour` and click *Initiate For All Recipients*.

Results

All recipients of the task are instructed to execute the task immediately, the next time they poll the server. The frequency with which Agents poll the server is controlled by the *Administration polling interval* property of the Agent.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 14: Modifying Scheduled Synchronizations [page 291]

Next task: Lesson 16: Changing the Remote Schema [page 293]

1.5.9.16 Lesson 16: Changing the Remote Schema

In this lesson, you change the schema of the remote database. For the purposes of this tutorial, a schema change occurs whenever you change the remote schema name of the database. You are never forced to change the remote schema name, it is always left up to your discretion.

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

You should try to ensure that any remote task that you can execute against one remote database can be executed against any other remote database with the same remote schema name. You should change a database's remote schema name whenever you change the database in a way that would make a task fail or succeed. The only commands within a task that are affected by the state of the remote database are the **Synchronize** and **Execute SQL** commands.

Synchronize commands depend on the presence of synchronization profiles in the remote database, so you should always change remote schema names when you add or remove a synchronization profile.

Execute SQL commands depend on the state of many database objects that you would normally consider to be part of the schema. Some examples of changes that would affect **Execute SQL** commands, and hence require a remote schema name change, are adding or removing tables from the database, altering the definition of tables in the database, and adding or removing stored procedures.

Procedure

1. Return to the *Folders* view in SQL Central. Under *MobiLink 17*, expand **Central Admin Tutorial**, right-click *Remote Schema Names* and click *New* > *Remote Schema Name*.

   The *Create Remote Schema Name Wizard* appears.
2. Type **Tutorial Application v2.0** for the schema name and click *Finish*.
3. Create a new remote task. In the *Folders* view of SQL Central under **Central Admin Tutorial**, right-click *Remote Tasks* and click *New* > *Remote Task*. The *Create Remote Task Wizard* appears.
4. On the *Welcome* page, type **Schema Upgrade** in the *Name* field.
5. Check *This task requires or creates a remote database* and set the *Remote schema name* to **Tutorial Application v1.0**.
6. Check *This task updates the schema of the managed remote database* and set *New remote schema name* to **Tutorial Application v2.0**. Click *Finish*.
7. On the *Commands* tab, choose *Execute SQL* from the *Command Type* dropdown list. In the *SQL* field, type the following:

```
CREATE TABLE  product (
    prod_id            integer primary key,
    name               varchar( 100 )
);
```

   The schema change task is now complete.

   Before you deploy the new schema change task, you must consider any tasks already assigned to the remote device. After the **Schema Upgrade** task completes, the remote schema name for the database is **Tutorial Application v2.0**. Any tasks on the remote device that are associated with the old remote schema name, **Tutorial Application v1.0** can no longer run and are discarded by the Agent. To maintain the functionality provided by these tasks, you must create new versions of the tasks and associate them with the new remote schema name.
8. In the *Folders* view under **Central Admin Tutorial**, expand *Consolidated Databases* > *Tutorial* > *Agents*, and then click **AID_JOHN**. Select the *Tasks* tab in the right pane. Only active tasks are still being executed by the Agent. These are the only tasks that you may need to create new versions of. In this case, the only active task is the **Sync every hour** task.

   You can determine if this task is associated with the old remote schema name by checking the *Remote Schema Name* column on the *Tasks* tab. This task shows that the *Remote Schema Name* of the **Sync every hour** task is **Tutorial Application v1.0**, so it is associated with the old remote schema name. To continue synchronization after the schema change, you need to create a new version of this task and assign it to the Agent.
9. Right-click the **Sync every hour** task and click *Go To Task*.
10. Right-click the deployed task **Sync every hour** and choose *Copy*.
11. Right-click *Remote Tasks* and click *Paste*. When you are asked for a name for the copied task, type **Sync every hour v2** and click *OK*.

12. Consider whether commands in the task require any changes to continue working with the new schema. In this case, the answer is no. There is only one command and it only depends on the `tutorial1_JOHN` synchronization profile, which you have not modified with this schema change.
13. Mark the task as being associated with the new remote schema name. Right-click the `Sync every hour v2` task and choose *Properties*. On the *General* tab of the properties window, choose `Tutorial Application v2.0` for the *Remote schema name* and click *OK*.
14. To deploy the new task, right-click the `Sync every hour v2` task and click *Deploy*. Click *Next*.
15. For *Recipients*, click *Specific agents* and then select agent `AID_JOHN`. Click *Next* and then click *Finish*.
16. Right-click the `Schema Upgrade` task and click *Deploy*. Click *Next*.
17. From the *Recipients* dropdown list, click *Specific agents* and assign the task to agent `AID_JOHN`. Click *Next* and then click *Finish*.

Results

You should see the `Schema Upgrade` task execute successfully. After that, the `Sync every hour v2` task should start executing each hour and the `Sync every hour` task should stop executing.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 15: Forcing Immediate Synchronization [page 292]

Next task: Lesson 17: Querying the Remote Database [page 295]

1.5.9.17 Lesson 17: Querying the Remote Database

In this lesson, you query the remote database and return results to the server. This is very useful when troubleshooting because you can find out exactly what state the remote database is in.

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

The tables you have added to the database in this tutorial do not contain any data, so instead you query the database system tables. Even though you are querying a system table in this example, everything you do works exactly the same way as if you queried a user table.

Suppose that you wanted to confirm that the schema change you performed in the last lesson did what you expected, that the product table was created with the correct columns. You could confirm that by querying the systable and systabcol system tables.

Procedure

1. In the *Folders* view of SQL Central under *MobiLink 17*, expand *Central Admin Tutorial*, right-click *Remote Tasks* and click *New* > *Remote Task*.

   The *Create Remote Task Wizard* appears.
2. On the *Welcome* page, type **`Table Query`** in the *Name* field.
3. Check *This task requires or creates a remote database* and set the *Remote schema name* to **`Tutorial Application v2.0`** and click *Next*.
4. On the *Trigger Mechanisms* page, check *When it is received by an agent* and click *Finish*.
5. Add an *Execute SQL* command to the task with the following SQL:

   ```
   SELECT * FROM SYS.SYSTAB WHERE table_name = 'product'
   go
   SELECT * FROM SYS.SYSTABCOL ORDER BY table_id
   ```
6. Right-click the new **`Table Query`** task and click *Deploy*. Click *Next*.
7. For *Recipients*, choose *Specific agents*, select agent **`AID_JOHN`** and click *Next* and then click *Next* again.
8. On the *Reporting Results And Status* page, set both *If task succeeds* and *If task fails* to *Send results and status immediately*. Click *Finish* and wait until the task executes.
9. Click the deployed copy of the **`Table Query`** task in the *Folders* view and then click the *Results* tab. If you don't see any results on the tab, wait until the tab is automatically refreshed or press F5 to refresh immediately.
10. Right-click the line in the table for the *Execute SQL* statement and choose *Details*.

    The *Command Result* window appears.
11. Click the *Results* tab on the window. This tab shows results of any queries executed. The *Result* dropdown at the top of the pane allows you to switch between results for the two queries. Click *Close*.

Results

The results from the remote database are displayed.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 16: Changing the Remote Schema [page 293]

Next task: Lesson 18: Uploading Files Using SIRT [page 297]

1.5.9.18 Lesson 18: Uploading Files Using SIRT

In this lesson, you upload files from the remote device using a server-initiated remote task (SIRT). Uploading files from the remote device is useful for troubleshooting because an administrator can examine the files for problems.

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

When you started the MobiLink Agent on the remote device, you directed it to log messages to the file `agent.txt`. You are now going to retrieve and examine that file from the remote device.

Procedure

1. In the *Folders* view of SQL Central under *MobiLink 17*, expand *Central Admin Tutorial*, right-click *Remote Tasks* and click *New* > *Remote Task*.

   The *Create Remote Task Wizard* appears.
2. On the *Welcome* page, type **`Upload Agent Log`** in the *Name* field.
3. Clear *This task requires or creates a remote database* if it is selected and click *Finish*.
4. Click the new task in the *Folders* view and add a command to the task. Set the *Command type* to *Upload file*.
5. Set the *Server file name* to `{agent_id}\agent.txt` and the *Remote file name* to `{agent_log}`. You can use the ellipsis (three dots) button in the command editor to easily enter the macro values.

The `{agent_log}` macro is replaced by the name of the log file being kept by the MobiLink Agent on the remote device.

In the *Server file name* field you specified the directory where the file is located using the `{agent_id}` macro. This is very important. If you do not use a macro when specifying the server file name, then every Agent that executes the task places their upload file in the same place, with each new Agent overwriting the file written by the previous agent. Using a macro ensures that each Agent uploads its log file to a different location on the server, allowing you to view all the Agent log files.

6. Right-click the new **`Upload Agent Log`** task and click *Deploy*. Click *Next*.
7. For *Recipients*, click *Specific agents* and then select agent **`AID_JOHN`**. Click *Next*.
8. On the *Delivery Options* page, click *The next time the agent synchronizes* and click *Next*.
9. On the *Reporting Results And Status* page, set both *If task succeeds* and *If task fails* to *Send results and status immediately*. Click *Finish*.
10. The task needs to be initiated by the administrator in SQL Central. To initiate the task, go to `AID_JOHN` under *Agents*. In the pane, select the *Tasks* tab, right-click the *Upload Agent Log* task and click *Initiate*. Wait for the task to execute.

Results

The uploaded file is placed in the MobiLink upload directory that was specified it with the -ftru option on the MobiLink command line. You specified `c:\cadmin_demo\consolidated\upload` for the upload directory. Take a look at that directory using a command prompt or the Windows Explorer. You should find the `AID_JOHN` subdirectory. In that subdirectory is the `agent.txt` file that you uploaded.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 17: Querying the Remote Database [page 295]

Next task: Lesson 19: Cleaning Up [page 299]

1.5.9.19 Lesson 19: Cleaning Up

Remove all tutorial materials from your computer.

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. Close any Interactive SQL, SQL Central, MobiLink, and synchronization client windows by right-clicking each task bar item and choosing Close.
2. Delete all tutorial-related data sources:
   a. Start the ODBC Data Source Administrator.
   b. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *ODBC Data Source Administrator*.
   c. Select *cadmin_tutorial_consol* from the list of *User Data Sources*, and click *Remove*.

Results

The tutorial resources are removed.

Task overview: Tutorial: Using Central Administration of Remote Databases [page 269]

Previous task: Lesson 18: Uploading Files Using SIRT [page 297]

1.5.10 Tutorial: Changing a Schema Using the Script Version Clause

This tutorial describes how to perform a schema change on a remote database involved in synchronization where the dbmlsync ScriptVersion extended option is not being used.

Prerequisites

This tutorial assumes you have a complete install of SQL Anywhere, including MobiLink on your local computer where you are running the tutorial.

You must have the following roles and privileges on the consolidated database:

- SYS_AUTH_RESOURCE_ROLE compatibility role
- MONITOR system privilege

You must have the following roles and privileges on the remote database:

- SYS_REPLICATION_ADMIN_ROLE system role
- SYS_RUN_REPLICATION_ROLE system role

Context

In this tutorial, you set up a synchronization system that synchronizes a single table, and then make a schema change to add a column to the synchronizing table and continue synchronizing.

This tutorial shows you how to:

- Create and configure the consolidated database
- Create and configure the remote database
- Synchronize the remote database
- Insert data in the remote database
- Perform a schema change on the consolidated database
- Perform a schema change on the remote database
- Insert data in the remote database
- Synchronize

1. Lesson 1: Creating and Configuring the Consolidated Database [page 301]
   In this lesson, you set up a consolidated database for synchronization.
2. Lesson 2: Creating and Configuring the Remote Database [page 303]
   In this lesson, you set up a remote database for synchronization.
3. Lesson 3: Synchronizing the Remote Database [page 304]
   You should now have a working synchronization system set up. In this lesson, you test it by inserting some data and synchronizing.

4. Lesson 4: Inserting Data in the Remote Database [page 305]
   In this lesson, you insert data into the remote database to demonstrate that a schema change can proceed even if there are operations in the remote database that need to be uploaded.
5. Lesson 5: Performing a Schema Change on the Consolidated Database [page 306]
   In this lesson, you perform a schema change on the consolidated database.
6. Lesson 6: Performing a Schema Change on the Remote Database [page 308]
   In this lesson, you modify the remote database to add the new column to the customer table and to change the script version used to synchronize.
7. Lesson 7: Inserting Data [page 309]
   In this lesson, you insert some more data into the remote and consolidated databases using the new schema.
8. Lesson 8: Synchronizing [page 310]
   In this lesson, you synchronize again with the schema changes.

Related Information

Schema Changes in Remote MobiLink Clients

1.5.10.1 Lesson 1: Creating and Configuring the Consolidated Database

In this lesson, you set up a consolidated database for synchronization.

Prerequisites

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

Procedure

1. Run the following commands to create a consolidated database and start it running.

```
md c:\cons
cd c:\cons
dbinit -dba DBA,passwd consol.db
dbsrv17 consol.db
```

2. Run the following command to define an ODBC data source for the consolidated database.

```
dbdsn -w dsn_consol -y -c "UID=DBA;PWD=passwd;DBF=consol.db;SERVER=consol"
```

3. To use a database as a consolidated database, you must run a setup script that adds system tables, views, and stored procedures that are used by MobiLink. The following command sets up `consol.db` as a consolidated database.

   ```
   dbisql -c "DSN=dsn_consol" %SQLANY17%\MobiLink\setup\syncsa.sql
   ```

4. Open Interactive SQL and connect to `consol.db` using the dsn_consol ODBC data source.

   ```
   dbisql -c "DSN=dsn_consol"
   ```

5. Execute the following SQL statements. They create the customer table on the consolidated database and create the required synchronization scripts.

   ```
   CREATE TABLE customer (
      id      unsigned integer primary key,
      name    varchar( 256),
      phone   varchar( 12 )
   );
   CALL ml_add_column('my_ver1', 'customer', 'id', null );
   CALL ml_add_column('my_ver1', 'customer', 'name', null );
   CALL ml_add_column('my_ver1', 'customer', 'phone', null );
   CALL ml_add_table_script( 'my_ver1', 'customer', 'upload_insert',
         'INSERT INTO customer ( id, name, phone ) '
         || 'VALUES ({ml r.id}, {ml r.name}, {ml r.phone} )' );
   CALL ml_add_table_script( 'my_ver1', 'customer', 'download_cursor',
          'SELECT id, name, phone from customer' );
   CALL ml_add_table_script( 'my_ver1', 'customer', 'download_delete_cursor', '--
   {ml_ignore}' );
   COMMIT;
   ```

   After you have executed the SQL, leave Interactive SQL running and connected to the database as you will be executing more SQL against the database as you work through the tutorial.

6. Start the MobiLink server by running the following command.

   ```
   start mlsrv17 -c "DSN=dsn_consol" -v+ -ot mlsrv.txt -zu+
   ```

Results

The consolidated database is created and set up to work with MobiLink.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Changing a Schema Using the Script Version Clause [page 300]

Next task: Lesson 2: Creating and Configuring the Remote Database [page 303]

1.5.10.2 Lesson 2: Creating and Configuring the Remote Database

In this lesson, you set up a remote database for synchronization.

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 commands to create a remote database and start it running.

   ```
   cd..
   md c:\remote
   cd c:\remote
   dbinit -dba DBA,passwd remote.db
   dbsrv17 remote.db
   ```

2. Open another instance of Interactive SQL and connect to `remote.db`.

   ```
   dbisql -c "SERVER=remote;DBF=remote.db;UID=DBA;PWD=passwd"
   ```

3. Execute the following SQL statement in Interactive SQL to create the table to be synchronized.

   ```
   CREATE TABLE customer (
      id       UNSIGNED INTEGER PRIMARY KEY,
      name     VARCHAR( 256),
      phone    VARCHAR( 12 )
   );
   ```

4. Still using the Interactive SQL instance connected to the remote database, create a publication, MobiLink user, and subscription. The script version is associated with the subscription using the SCRIPT VERSION clause. This is very important since the schema upgrade procedure shown in this tutorial only works for subscriptions that have the script version set using the SCRIPT VERSION clause.

   ```
   CREATE PUBLICATION p1 (
         TABLE customer
   );
   CREATE SYNCHRONIZATION USER u1;
   CREATE SYNCHRONIZATION SUBSCRIPTION my_sub
   TO p1
   FOR u1
   SCRIPT VERSION 'my_ver1';
   ```

   After you have executed the SQL, leave Interactive SQL running and connected to the database as you will be running more SQL against the database as you work through the tutorial.

Results

The remote database is created and configured.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Changing a Schema Using the Script Version Clause [page 300]

Previous task: Lesson 1: Creating and Configuring the Consolidated Database [page 301]

Next task: Lesson 3: Synchronizing the Remote Database [page 304]

1.5.10.3 Lesson 3: Synchronizing the Remote Database

You should now have a working synchronization system set up. In this lesson, you test it by inserting some data and synchronizing.

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. Using the instance of Interactive SQL that is connected to the consolidated database, execute the following SQL statement to insert a row in the customer table.

```
INSERT INTO customer VALUES( 100, 'John Jones', '519-555-1234' );
COMMIT;
```

2. Using the instance of Interactive SQL that is connected to the remote database, execute the following SQL statement to insert a row in the customer table.

```
INSERT INTO customer VALUES( 1, 'Willie Lowman', '705-411-6372' );
COMMIT;
```

3. Synchronize by running the following command.

```
dbmlsync -v+ -ot sync1.txt -c UID=DBA;PWD=passwd;SERVER=remote -s my_sub -k
```

Results

The remote database is synchronized.

You can confirm that the synchronization succeeded by comparing the contents of the customer table in the remote and consolidated databases. You might also want to look at the dbmlsync log, `sync1.txt` and check for errors.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Changing a Schema Using the Script Version Clause [page 300]

Previous task: Lesson 2: Creating and Configuring the Remote Database [page 303]

Next task: Lesson 4: Inserting Data in the Remote Database [page 305]

1.5.10.4 Lesson 4: Inserting Data in the Remote Database

In this lesson, you insert data into the remote database to demonstrate that a schema change can proceed even if there are operations in the remote database that need to be uploaded.

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

Using the instance of Interactive SQL that is connected to the remote database, execute the following SQL statement to insert a row in the customer table.

```
INSERT INTO customer VALUES( 2, 'Sue Slow', '602-411-5467' );
COMMIT;
```

Results

The data is inserted into the remote database.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Changing a Schema Using the Script Version Clause [page 300]

Previous task: Lesson 3: Synchronizing the Remote Database [page 304]

Next task: Lesson 5: Performing a Schema Change on the Consolidated Database [page 306]

1.5.10.5 Lesson 5: Performing a Schema Change on the Consolidated Database

In this lesson, you perform a schema change on the consolidated 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. Add a new column to the customer table to store the customer's cell phone number. First, add the new column to the consolidated database by executing the following SQL statement in the instance of Interactive SQL that is connected to the consolidated database.

   ```
   ALTER TABLE customer ADD cell_phone VARCHAR(12) DEFAULT NULL;
   ```

2. Create a new script version called **`my_ver2`** to handle synchronizations from remote databases with the new schema. Remote databases with the old schema continue to use the old script version, **`my_ver1`**. Execute the following SQL statements on the consolidated database.

   ```
   CALL ml_add_column('my_ver2', 'customer', 'id', null );
   CALL ml_add_column('my_ver2', 'customer', 'name', null );
   CALL ml_add_column('my_ver2', 'customer', 'phone', null );
   CALL ml_add_column('my_ver2', 'customer', 'cell_phone', null );
   CALL ml_add_table_script( 'my_ver2', 'customer', 'upload_insert',
         'INSERT INTO customer ( id, name, phone, cell_phone ) '
         || 'VALUES ({ml r.id}, {ml r.name}, {ml r.phone}, {ml r.cell_phone})' );
   CALL ml_add_table_script( 'my_ver2', 'customer', 'download_cursor',
          'SELECT id, name, phone, cell_phone from customer' );
   CALL ml_add_table_script( 'my_ver2', 'customer', 'download_delete_cursor', '--
   {ml_ignore}' );
   COMMIT;
   ```

Results

The consolidated database schema is updated.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Changing a Schema Using the Script Version Clause [page 300]

Previous task: Lesson 4: Inserting Data in the Remote Database [page 305]

Next task: Lesson 6: Performing a Schema Change on the Remote Database [page 308]

1.5.10.6 Lesson 6: Performing a Schema Change on the Remote Database

In this lesson, you modify the remote database to add the new column to the customer table and to change the script version used to synchronize.

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 a synchronization schema change. This is required for most schema changes that affect synchronizing tables. This statement changes the script version that is used to synchronize the subscription, and locks the affected table so the schema change can proceed safely.

   Execute the following SQL statement on the remote database using the instance of Interactive SQL that is connected to the remote database.

   ```
   START SYNCHRONIZATION SCHEMA CHANGE
   FOR TABLES customer
   SET SCRIPT VERSION = 'my_ver2';
   ```

2. Add the new column to the customer table by executing the following SQL statement.

   ```
   ALTER TABLE customer ADD cell_phone VARCHAR(12) DEFAULT NULL;
   ```

3. Close the schema change, which unlocks the tables.

   ```
   STOP SYNCHRONIZATION SCHEMA CHANGE;
   ```

Results

A synchronization schema change is performed on the remote database.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Changing a Schema Using the Script Version Clause [page 300]

Previous task: Lesson 5: Performing a Schema Change on the Consolidated Database [page 306]

Next task: Lesson 7: Inserting Data [page 309]

1.5.10.7 Lesson 7: Inserting Data

In this lesson, you insert some more data into the remote and consolidated databases using the new schema.

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. Using Interactive SQL, execute the following SQL statements on the remote database.

```
INSERT INTO customer VALUES( 3, 'Mo Hamid', '613-411-9999', '613-502-1212' );
COMMIT;
```

2. Using Interactive SQL, execute the following SQL statements on the consolidated database.

```
INSERT INTO customer VALUES( 101, 'Theo Tug', '212-911-7677',
'212-311-3900' );
COMMIT;
```

Results

Data is inserted into the remote and consolidated databases using the new schema.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Changing a Schema Using the Script Version Clause [page 300]

Previous task: Lesson 6: Performing a Schema Change on the Remote Database [page 308]

Next task: Lesson 8: Synchronizing [page 310]

1.5.10.8 Lesson 8: Synchronizing

In this lesson, you synchronize again with the schema changes.

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

Synchronize again by running the following command:

```
dbmlsync -v+ -ot sync2.txt -c UID=DBA;PWD=passwd;SERVER=remote -s my_sub -k
```

The row for **`Sue Slow`** that was inserted before the schema change is uploaded using the script version **`my_ver1`**. The row for **`Mo Hamid`** that was inserted after the schema change is uploaded using the script version **`my_ver2`**. Rows are downloaded using the download cursor for **`my_ver2`**.

Results

The schema change is now complete and you can continue synchronizing normally.

Task overview: Tutorial: Changing a Schema Using the Script Version Clause [page 300]

Previous task: Lesson 7: Inserting Data [page 309]

1.5.11 Tutorial: Changing a Schema Using the ScriptVersion Extended Option

This tutorial demonstrates how to perform a schema change when you are using the ScriptVersion extended option.

Prerequisites

This tutorial assumes you have a complete install of SQL Anywhere, including MobiLink on your local computer where you are running the tutorial.

You must have the following roles and privileges on the consolidated database:

- SYS_AUTH_RESOURCE_ROLE compatibility role
- MONITOR system privilege

You must have the following roles and privileges on the remote database:

- SYS_REPLICATION_ADMIN_ROLE system role
- SYS_RUN_REPLICATION_ROLE system role

Context

This tutorial shows you how to:

- Create and configure the consolidated database
- Create and configure the remote database
- Synchronize the remote database
- Perform a schema change on the consolidated database
- Perform a schema change on the remote database

i Note

Avoid using the ScriptVersion extended option if possible. Instead, associate your script version with your subscription using the SCRIPT VERSION clause of the CREATE SYNCHRONIZATION SUBSCRIPTION statement or the SET SCRIPT VERSION clause of the ALTER SYNCHRONIZATION SUBSCRIPTION statement. These implementations give you more flexibility to perform schema upgrades.

1. Lesson 1: Creating and Configuring the Consolidated Database [page 312]
   In this lesson, you set up a consolidated database for synchronization.
2. Lesson 2: Creating and Configuring the Remote Database [page 313]
   In this lesson, you set up a remote database for synchronization.
3. Lesson 3: Synchronizing the Remote Database [page 315]
   You should now have a working synchronization system set up. In this lesson, you test it by inserting some data and synchronizing.

4. Lesson 4: Performing a Schema Change on the Consolidated Database [page 316]
   In this lesson, you add a new column to the customer table to store the customer's cell phone number.
5. Lesson 5: Performing a Schema Change on the Remote Database [page 317]
   In this lesson, you modify the remote database to add the new column to the customer table and to change the script version used to synchronize.

1.5.11.1 Lesson 1: Creating and Configuring the Consolidated Database

In this lesson, you set up a consolidated database for synchronization.

Prerequisites

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

Procedure

1. Run the following commands to create and start a consolidated database.

   ```
   md c:\cons
   cd c:\cons
   dbinit -dba DBA,passwd consol.db
   dbsrv17 consol.db
   ```
2. Run the following command to define an ODBC data source for the consolidated database:

   ```
   dbdsn -w dsn_consol -y -c "UID=DBA;PWD=passwd;DBF=consol.db;SERVER=consol"
   ```
3. To use a database as a consolidated database, you must run a setup script that adds system tables, views, and stored procedures that are used by MobiLink. The following command sets up `consol.db` as a consolidated database.

   ```
   dbisql -c "DSN=dsn_consol" %SQLANY17%\MobiLink\setup\syncsa.sql
   ```
4. Open Interactive SQL and connect to `consol.db` using the **`dsn_consol`** DSN.

   ```
   dbisql -c "DSN=dsn_consol"
   ```
5. Execute the following SQL statements in Interactive SQL. They create the customer table on the consolidated database and create the required synchronization scripts.

   ```
   CREATE TABLE customer (
      id      unsigned integer primary key,
      name    varchar( 256),
      phone   varchar( 12 )
   );
   CALL ml_add_column('my_ver1', 'customer', 'id', null );
   CALL ml_add_column('my_ver1', 'customer', 'name', null );
   ```

```
CALL ml_add_column('my_ver1', 'customer', 'phone', null );
CALL ml_add_table_script( 'my_ver1', 'customer', 'upload_insert',
      'INSERT INTO customer ( id, name, phone ) '
      || 'VALUES ({ml r.id}, {ml r.name}, {ml r.phone} )' );
CALL ml_add_table_script( 'my_ver1', 'customer', 'download_cursor',
       'SELECT id, name, phone from customer' );
CALL ml_add_table_script( 'my_ver1', 'customer', 'download_delete_cursor', '--
{ml_ignore}' );
COMMIT;
```

After you have executed the SQL statements, leave Interactive SQL running and connected to the database as you will be executing more SQL against the database as you work through the tutorial.

6. Start the MobiLink server by running the following command:

```
start mlsrv17 -c "DSN=dsn_consol" -v+ -ot mlsrv.txt -zu+
```

Results

The consolidated database is created and configured for synchronization.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Changing a Schema Using the ScriptVersion Extended Option [page 311]

Next task: Lesson 2: Creating and Configuring the Remote Database [page 313]

1.5.11.2 Lesson 2: Creating and Configuring the Remote Database

In this lesson, you set up a remote database for synchronization.

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 commands to create and start a remote database.

```
cd..
md c:\remote
cd c:\remote
dbinit -dba DBA,passwd remote.db
dbeng17 remote.db
```

2. Open another instance of Interactive SQL and connect to `remote.db`.

```
dbisql -c "SERVER=remote;DBF=remote.db;UID=DBA;PWD=passwd"
```

3. Execute the following SQL statement in Interactive SQL to create the table to be synchronized.

```
CREATE TABLE customer (
   id      unsigned integer primary key,
   name    varchar( 256),
   phone   varchar( 12 )
);
```

4. Create a publication, MobiLink user, and subscription.

```
CREATE PUBLICATION p1 (
   TABLE customer
);
CREATE SYNCHRONIZATION USER u1;
CREATE SYNCHRONIZATION SUBSCRIPTION my_sub
TO p1
FOR u1
OPTION ScriptVersion='my_ver1';
```

   After you have executed the SQL statements, leave Interactive SQL running and connected to the database as you will be running more SQL statements on the database as you work through the tutorial.

Results

The remote database is created and configured for synchronization.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Changing a Schema Using the ScriptVersion Extended Option [page 311]

Previous task: Lesson 1: Creating and Configuring the Consolidated Database [page 312]

Next task: Lesson 3: Synchronizing the Remote Database [page 315]

1.5.11.3 Lesson 3: Synchronizing the Remote Database

You should now have a working synchronization system set up. In this lesson, you test it by inserting some data and synchronizing.

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. Using the instance of Interactive SQL that is connected to the consolidated database, execute the following SQL statements to insert a row in the customer table.

```
INSERT INTO customer VALUES( 100, 'John Jones', '519-555-1234' );
COMMIT;
```

2. Using the instance of Interactive SQL that is connected to the remote database, execute the following SQL statements to insert a row in the customer table.

```
INSERT INTO customer VALUES( 1, 'Willie Lowman', '705-411-6372' );
COMMIT;
```

3. Synchronize by running the following command.

```
dbmlsync -v+ -ot sync1.txt -c UID=DBA;PWD=passwd;SERVER=remote -s my_sub -k
```

   You can confirm that the synchronization succeeded by comparing the contents of the customer table in the remote and consolidated databases. You might also want to look at the dbmlsync log, `sync1.txt` and check for errors.

Results

The consolidated and remote databases are synchronized.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Changing a Schema Using the ScriptVersion Extended Option [page 311]

Previous task: Lesson 2: Creating and Configuring the Remote Database [page 313]

Next task: Lesson 4: Performing a Schema Change on the Consolidated Database [page 316]

1.5.11.4 Lesson 4: Performing a Schema Change on the Consolidated Database

In this lesson, you add a new column to the customer table to store the customer's cell phone number.

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. Using the instance of Interactive SQL that is connected to the consolidated database, execute the following SQL statement to insert a row in the customer table.

```
ALTER TABLE customer ADD cell_phone VARCHAR(12) DEFAULT NULL;
```

2. Create a new script version called my_ver2 to handle synchronizations from remote databases with the new schema. Remote databases with the old schema continue to use the old script version, my_ver1.

   Execute the following SQL statements on the consolidated database:

```
CALL ml_add_column('my_ver2', 'customer', 'id', null );
CALL ml_add_column('my_ver2', 'customer', 'name', null );
CALL ml_add_column('my_ver2', 'customer', 'phone', null );
CALL ml_add_column('my_ver2', 'customer', 'cell_phone', null );
CALL ml_add_table_script( 'my_ver2', 'customer', 'upload_insert',
      'INSERT INTO customer ( id, name, phone, cell_phone ) '
      || 'VALUES ({ml r.id}, {ml r.name}, {ml r.phone}, {ml r.cell_phone})' );
CALL ml_add_table_script( 'my_ver2', 'customer', 'download_cursor',
       'SELECT id, name, phone, cell_phone from customer' );
CALL ml_add_table_script( 'my_ver2', 'customer', 'download_delete_cursor', '--
{ml_ignore}' );
COMMIT;
```

Results

Changes are made to the consolidated database and a new script version is created to handle the schema change.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Changing a Schema Using the ScriptVersion Extended Option [page 311]

Previous task: Lesson 3: Synchronizing the Remote Database [page 315]

Next task: Lesson 5: Performing a Schema Change on the Remote Database [page 317]

1.5.11.5 Lesson 5: Performing a Schema Change on the Remote Database

In this lesson, you modify the remote database to add the new column to the customer table and to change the script version used to synchronize.

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.

Before you modify the remote database and change the script version used to synchronize, you must ensure that there are no operations for the customer table that need to be uploaded. The best way to do this is to perform the schema change in the sp_hook_dbmlsync_schema_upgrade hook. When you use this hook, dbmlsync ensures that the schema change is performed safely by locking the synchronizing tables at the start of synchronization and holding the locks until the schema change is complete.

Context

> ⚠ Caution
>
> If you change the schema when there are operations to be uploaded, the remote database is always unable to synchronize after the schema change.

Procedure

1. Create an sp_hook_dbmlsync_schema_upgrade hook by executing the following SQL statement on the remote database. The hook adds a new column to the customer table and changes the value of the ScriptVersion extended option stored with the subscription. The hook is deleted by dbmlsync after it has executed.

   ```
   CREATE PROCEDURE sp_hook_dbmlsync_schema_upgrade()
   BEGIN
       ALTER TABLE customer
       ADD cell_phone varchar(12) default null;
       ALTER SYNCHRONIZATION SUBSCRIPTION my_sub
       ALTER OPTION ScriptVersion='my_ver2';
       UPDATE #hook_dict
       SET value = 'always'
           WHERE name = 'drop hook';
   END;
   ```

2. Synchronize to upload any operations that need to be uploaded and to perform the schema change by executing the sp_hook_dbmlsync_schema_change hook. Run the following command.

   ```
   dbmlsync -v+ -ot sync2.txt -c UID=DBA;PWD=passwd;SERVER=remote -s my_sub -k
   ```

   After this synchronization, it is a very good idea to look at the dbmlsync log `sync2.txt` to ensure that there are no errors to indicate that the schema change was not completed.

Results

The schema change is now complete and you can continue synchronizing normally.

Task overview: Tutorial: Changing a Schema Using the ScriptVersion Extended Option [page 311]

Previous task: Lesson 4: Performing a Schema Change on the Consolidated Database [page 316]

1.5.12 Tutorial: Simulating Multiple MobiLink Clients Using the MobiLink Replay Utility

This tutorial demonstrates how to use the mlreplay utility to simulate multiple MobiLink clients on a single computer.

Prerequisites

You require:

- Basic knowledge of MobiLink event scripts

The following software is required:

- SQL Anywhere 17

You must have the following roles and privileges on the consolidated database:

- SYS_AUTH_RESOURCE_ROLE compatibility role
- MONITOR system privilege

You must have the following roles and privileges on the remote database:

- SYS_REPLICATION_ADMIN_ROLE system role
- SYS_RUN_REPLICATION_ROLE system role

Context

This tutorial shows you how to:

- Set up a MobiLink consolidated database
- Start the MobiLink server to record and replay synchronizations
- Use the mlreplay utility to simulate MobiLink clients

synchronize with the consolidated database, and create a synchronization publication, user, and subscription.

6. Lesson 6: Recording Synchronization [page 329]
   In this lesson, you run the dbmlsync utility to initiate MobiLink synchronization for SQL Anywhere remote databases.
7. Lesson 7: Restarting the MobiLink Server to Replay [page 331]
   In this lesson, you stop the MobiLink server to stop recording and then restart the server without the -rp option to prepare the server for replay.
8. Lesson 8: Replaying Synchronization [page 333]
   In this lesson, you perform a synchronization so that the schema is cached on the MobiLink server. You create the simulated client information file to replay the MobiLink protocol information about the simulated clients.
9. Lesson 9: Cleaning Up [page 335]
   Remove the tutorials materials from your computer.

Related Information

MobiLink Replay Utility (mlreplay)

1.5.12.1 Lesson 1: Setting up Your MobiLink Consolidated Database

In this lesson, you set up your MobiLink consolidated database.

Prerequisites

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

Procedure

1. Create a new working directory to store all the sample files created in this tutorial. This tutorial assumes the path `c:\mlreplay`.
2. At a command prompt, change the working directory to `c:\mlreplay`. This tutorial assumes all commands are run from this directory.
3. Run the following command to create a SQL Anywhere consolidated database named `cons.db`:

```
dbinit -dba DBA,passwd cons.db
```

4. Run the following command to start the consolidated database:

```
dbsrv17 cons.db
```

5. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *ODBC Data Source Administrator*.
6. Click the *User DSN* tab, and click *Add*.
7. In the *Create New Data Source* window, click *SQL Anywhere 17* and click *Finish*.
8. Perform the following tasks in the *ODBC Configuration For SQL Anywhere* window:
    a. Click the *ODBC* tab.
    b. In the *Data source name* field, type **`cons`**.
    c. Click the *Login* tab.
    d. In the *User ID* field, type **`DBA`**.
    e. In the *Password* field, type **`passwd`**.
    f. From the *Action* dropdown list, choose *Connect to a running database on this computer*.
    g. In the *Server name* field, type **`cons`**.
    h. In the *Database name* field, type **`cons`**.
    i. Click *OK*.
9. Close ODBC data source administrator.

    Click *OK* on the *ODBC Data Source Administrator* window.
10. Connect to your consolidated database in Interactive SQL.

    Run the following command:

```
dbisql -c "DSN=cons"
```

11. Execute the following statement in Interactive SQL to create MobiLink system tables and stored procedures using the `syncsa.sql` setup script. Replace *C:\Program Files\SQL Anywhere 17*\ with the location of your SQL Anywhere 17 installation.

```
READ "C:\Program Files\SQL Anywhere 17\MobiLink\setup\syncsa.sql";
```

    Interactive SQL applies `syncsa.sql` to your consolidated database. Running `syncsa.sql` creates a series of system tables and stored procedures prefixed with **ml_**. The MobiLink server works with these tables and stored procedures in the synchronization process.
12. Execute the following SQL statement in Interactive SQL to create the **`T1`** table:

```
CREATE TABLE T1 (
    pk1     INTEGER,
    pk2     INTEGER,
    c1      VARCHAR(30000),
    PRIMARY KEY(pk1, pk2)
);
```

    Interactive SQL creates the **`T1`** table in your consolidated database.
13. Close Interactive SQL.

Results

The consolidated database is set up.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Simulating Multiple MobiLink Clients Using the MobiLink Replay Utility [page 319]

Next task: Lesson 2: Creating a MobiLink Project [page 322]

1.5.12.2 Lesson 2: Creating a MobiLink Project

In this lesson, you connect to the consolidated database by creating a new MobiLink project.

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. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *SQL Central*.
2. Click *Tools* > *MobiLink 17* > *New Project*.
3. In the *Name* field, type **`mlreplay_project`**.
4. In the *Location* field, type `C:\mlreplay`, and click *Next*.
5. In the *Database display name* field, type **`cons`**.
6. Click *Edit*. The *Connect to a generic ODBC database* window appears.
7. In the *User ID* field, type **`DBA`**.
8. In the *Password* field, type **`passwd`**.
9. In the *ODBC Data Source name* field, click *Browse*, and select **`cons`**.
10. Click *OK*, and click *Save*.
11. Check the *Remember the password* option, and click *Next*, accepting all the defaults, until you get to the end of the wizard.

12. Click *Finish*.

Results

The MobiLink project is created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Simulating Multiple MobiLink Clients Using the MobiLink Replay Utility [page 319]

Previous task: Lesson 1: Setting up Your MobiLink Consolidated Database [page 320]

Next task: Lesson 3: Adding Synchronization Scripts [page 323]

1.5.12.3 Lesson 3: Adding Synchronization Scripts

Each script belongs to a designated script version. You must add a script version to the consolidated database before you add scripts.

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

You can view, write, and modify synchronization scripts using SQL Central. In this lesson, you write the following synchronization scripts:

upload_insert

This event defines how new client-side data should be applied to the consolidated database.

download_cursor

This event defines the data that should be downloaded to remote clients.

download_delete_cursor

This event is required when using synchronization scripts that are not upload-only. You set the MobiLink server to ignore this event for the purpose of this tutorial.

Procedure

1. Click *View* > *Folders*.
2. In the left pane of SQL Central under *MobiLink 17*, expand **mlreplay_project**, *Consolidated Databases*, **cons - DBA**.
3. Right-click *Versions* and choose *New* > *Version*.
4. In the *What do you want to name the new script version* field, type **MLReplayDemo**.
5. Click *Finish*.
6. In the left pane of SQL Central under *MobiLink 17*, expand **mlreplay_project**, *Consolidated Databases*, **cons - DBA**.
7. Right-click *Synchronized Tables* and click *New* > *Synchronized Table*.
8. Click the *Choose a table in the consolidated database with the same name as the remote table* option.
9. In the *Which user owns the table you want to synchronize* list, click *DBA*.
10. In the *Which table do you want to synchronize* list, click *T1*.
11. Click *Finish*.

    The **T1** table is registered as a synchronization table and you can add scripts to that table.
12. In the left pane of SQL Central under *MobiLink 17*, expand **mlreplay_project**, *Consolidated Databases*, **cons - DBA**, *Synchronized Tables*.
13. Right-click *T1* and click *New* > *Table Script*.
14. In the *For which version do you want to create the table script* list, click **MLReplayDemo**.
15. In the *Which event should cause the table script to be executed* list, click *upload_insert* and click *Next*.
16. Click *Finish*.
17. In the right pane of SQL Central, use the following SQL script for the *upload_insert* event:

    ```
    INSERT INTO T1 VALUES( cast({ml s.remote_id} as INTEGER), {ml r.2}, {ml r.
    3} );
    ```

    The upload_insert event determines how data inserted into the remote database should be applied to the consolidated database.
18. Click *File* > *Save*.
19. Repeat steps 13 to 16, specifying the *download_cursor* event instead of the *upload_insert* event in step 15.
20. In the right pane of SQL Central, use the following SQL script for the *download_cursor* event:

    ```
    SELECT pk1, pk2, c1 FROM T1;
    ```

    The download_cursor script defines a cursor to select consolidated database rows that are downloaded and inserted or updated in the remote database.

21. Click ▶ *File* ▶ *Save* ▶.
22. Repeat steps 13 to 16, specifying the *download_delete_cursor* event instead of the *upload_insert* event in step 15.
23. In the right pane of SQL Central, use the following SQL script for the *download_delete_cursor* event:

    ```
    --{ml_ignore}
    ```

24. Click ▶ *File* ▶ *Save* ▶.

Results

The synchronization scripts are created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Simulating Multiple MobiLink Clients Using the MobiLink Replay Utility [page 319]

Previous task: Lesson 2: Creating a MobiLink Project [page 322]

Next task: Lesson 4: Starting the MobiLink Server to Record [page 326]

Related Information

Overview of MobiLink Events
Script Additions and Deletions
Scripts to Upload Rows
Scripts to Download Rows
Direct Row Handling
Direct Uploads
Direct Downloads
Partitioned Rows Among Remote Databases
Implementing Timestamp-based Downloads
upload_insert Table Event
upload_update Table Event
upload_delete Table Event
download_cursor Table Event
download_delete_cursor Table Event

1.5.12.4 Lesson 4: Starting the MobiLink Server to Record

In this lesson, you start the MobiLink server (mlsrv17) using the -c option to connect to your consolidated 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

Run the following command to connect to your consolidated database:

```
mlsrv17 -c "DSN=cons" -zu+ -zs mlreplay_svr -x tcpip -ot mlsrv.mls -v+ -rp .
```

Below is a description of each MobiLink server option used. The -ot and -v options provide debugging and troubleshooting information. These logging options are appropriate in a development environment. Typically, for performance reasons, -v is not used in production.

Option	Description
-c	Precedes the connection string.
-ot	Specifies the message log file `mlsrv.mls`.
-v+	Specifies what information is logged. Using -v+ sets maximum verbose logging.
-rp	Specifies the directory where synchronizations are recorded for playback.
-x	Sets the protocol used to listen for synchronization requests.
-zs	Sets a MobiLink server name.
-zu+	Adds new users automatically.

Results

The MobiLink server started and connected to the consolidated database. The MobiLink server messages window appears.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Simulating Multiple MobiLink Clients Using the MobiLink Replay Utility [page 319]

Previous task: Lesson 3: Adding Synchronization Scripts [page 323]

Next task: Lesson 5: Setting up Your MobiLink Client Database [page 327]

Related Information

MobiLink Server Options

1.5.12.5 Lesson 5: Setting up Your MobiLink Client Database

MobiLink is designed for synchronization involving a consolidated database server and a large number of mobile databases. In this lesson, you create a remote database, create a `T1` table, which you synchronize with the consolidated database, and create a synchronization publication, user, and subscription.

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

In this lesson, you use a SQL Anywhere database for your consolidated database and your MobiLink client. For tutorial purposes, your MobiLink client, consolidated database, and MobiLink server all reside on the same computer.

To set up the MobiLink client database, you create a `T1` table for the remote database. The `T1` table corresponds to the `T1` table on the consolidated database. The MobiLink server uses SQL-based scripts to synchronize product quantities.

You create a synchronization user, publication, and subscription on the client database after creating the tables. Publications identify the tables and columns on your remote database that you want synchronized. These tables and columns are called **articles**. A synchronization subscription subscribes a MobiLink user to a publication.

Procedure

1. Create your MobiLink client databases using the dbinit command line utility.

   Run the following command to create the **`remote`** database:

   ```
   dbinit -dba DBA,passwd remote.db
   ```
2. Start your MobiLink client database using the dbeng17 command line utility.

   Run the following command to start the **`remote`** database:

   ```
   dbeng17 remote
   ```
3. Connect to the remote database using Interactive SQL.

   Run the following command:

   ```
   dbisql -c "SERVER=remote;UID=DBA;PWD=passwd"
   ```
4. Create the **`T1`** table for the **`remote`** database.

   Execute the following SQL statements in Interactive SQL:

   ```
   CREATE TABLE T1 (
       pk1      INTEGER,
       pk2      INTEGER,
       c1       VARCHAR(30000),
       PRIMARY KEY(pk1,pk2)
   );
   SET OPTION PUBLIC.ml_remote_id = '0';
   ```
5. Create your MobiLink synchronization user, publication, and subscription for the **`remote`** database.

   Execute the following SQL statement in Interactive SQL:

   ```
   CREATE PUBLICATION P1 ( TABLE T1 );
   CREATE SYNCHRONIZATION USER U1;
   CREATE SYNCHRONIZATION SUBSCRIPTION TO P1 FOR U1 TYPE 'TCPIP' ADDRESS
   'host=localhost;port=2439';
   ```
6. Keep Interactive SQL open for the next lesson.

Results

A remote database, **`T1`** table, synchronization publication, user, and subscription are created.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Simulating Multiple MobiLink Clients Using the MobiLink Replay Utility [page 319]

Previous task: Lesson 4: Starting the MobiLink Server to Record [page 326]

Next task: Lesson 6: Recording Synchronization [page 329]

Related Information

MobiLink Clients
Initialization Utility (dbinit)
CREATE SYNCHRONIZATION USER Statement [MobiLink]
CREATE PUBLICATION Statement [MobiLink] [SQL Remote]
CREATE SYNCHRONIZATION SUBSCRIPTION Statement [MobiLink]

1.5.12.6 Lesson 6: Recording Synchronization

In this lesson, you run the dbmlsync utility to initiate MobiLink synchronization for SQL Anywhere remote databases.

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. Perform the first recorded synchronization so that the schema is cached on the MobiLink server.

   Run the following command to synchronize the **remote** database:

   ```
   dbmlsync -c "SERVER=remote;UID=DBA;PWD=passwd" -ot remote1.mls -e
   "sv=MLReplayDemo" -v+
   ```

   The following table contains a description for each dbmlsync option used:

Option	Description
-c	Specifies the connection string.
-ot	Specifies the file to log messages in.
-e	Specifies the script version to synchronize with.

Option	Description
-v+	Specifies what information is logged. Using -v+ sets maximum verbose logging.

An output screen appears indicating that the synchronization succeeded. SQL-based synchronization transferred rows in the client `T1` table to the `T1` table in the consolidated database.

2. Prepare the remote database for data insertion so that a second synchronization occurs.

   You should still be connected to the *remote* database with Interactive SQL. If you are not, run the following command to connect to the `remote` database:

   ```
   dbisql -c "SERVER=remote;UID=DBA;PWD=passwd"
   ```

3. Load data into the `remote` database to be uploaded to the MobiLink server by during the replay session.

   Execute the following SQL statement in Interactive SQL:

   ```
   INSERT INTO T1 (pk1,pk2,c1) values (0,1,'data1');
   INSERT INTO T1 (pk1,pk2,c1) values (0,2,'data2');
   INSERT INTO T1 (pk1,pk2,c1) values (0,3,'data3');
   INSERT INTO T1 (pk1,pk2,c1) values (0,4,'data4');
   INSERT INTO T1 (pk1,pk2,c1) values (0,5,'data5');
   INSERT INTO T1 (pk1,pk2,c1) values (0,6,'data6');
   INSERT INTO T1 (pk1,pk2,c1) values (0,7,'data7');
   INSERT INTO T1 (pk1,pk2,c1) values (0,8,'data8');
   INSERT INTO T1 (pk1,pk2,c1) values (0,9,'data9');
   INSERT INTO T1 (pk1,pk2,c1) values (0,10,'data10');
   COMMIT;
   ```

4. Perform the second recorded synchronization. This is the protocol that gets replayed.

   Run the following command to synchronize the `remote` database:

   ```
   dbmlsync -c "SERVER=remote;UID=DBA;PWD=passwd" -ot remote2.mls -e
   "sv=MLReplayDemo" -v+
   ```

Results

The databases are synchronized.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Simulating Multiple MobiLink Clients Using the MobiLink Replay Utility [page 319]

Previous task: Lesson 5: Setting up Your MobiLink Client Database [page 327]

Next task: Lesson 7: Restarting the MobiLink Server to Replay [page 331]

Related Information

SQL Anywhere Clients
MobiLink SQL Anywhere Client Utility (dbmlsync) Syntax

1.5.12.7 Lesson 7: Restarting the MobiLink Server to Replay

In this lesson, you stop the MobiLink server to stop recording and then restart the server without the -rp option to prepare the server for replay.

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 stop the MobiLink server, **`mlreplay_svr`**:

   ```
   mlstop -w -t 1m mlreplay_svr
   ```

   The following table contains a description for each option used:

Option	Description
-w	Waits for the server to shut down before returning to the command prompt.
-t	Specifies that the server should shut down after one minute, or after the current synchronizations have completed, whichever is sooner.

   The MobiLink server stops along with synchronization recording.
2. Run the following command to connect to your consolidated database:

   ```
   mlsrv17 -c "DSN=cons" -zu+ -zs mlreplay_svr -x tcpip -ot server_replay.mls -v+
   ```

   Below is a description of each MobiLink server option used. The -ot and -v options provide debugging and troubleshooting information. These logging options are appropriate in a development environment. Typically, for performance reasons, -v is not used in production.

Option	Description
-c	Specifies the connection string.
-ot	Specifies the message log file `server_replay.mls`.
-v+	Specifies what information is logged. Using -v+ sets maximum verbose logging.
-x	Sets the protocol used to listen for synchronization requests.
-zs	Sets a MobiLink server name.
-zu+	Adds new users automatically.

The MobiLink server messages window appears.

Results

The MobiLink server is stopped and then started again.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Simulating Multiple MobiLink Clients Using the MobiLink Replay Utility [page 319]

Previous task: Lesson 6: Recording Synchronization [page 329]

Next task: Lesson 8: Replaying Synchronization [page 333]

Related Information

MobiLink Server Options

1.5.12.8 Lesson 8: Replaying Synchronization

In this lesson, you perform a synchronization so that the schema is cached on the MobiLink server. You create the simulated client information file to replay the MobiLink protocol information about the simulated clients.

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

The simulated client information file is only necessary when replaying the recorded protocol concurrently across multiple simulated clients.

Procedure

1. Run the following command to synchronize the `remote` database:

   ```
   dbmlsync -c "SERVER=remote;UID=DBA;PWD=passwd" -ot remote3.mls -e
   "sv=MLReplayDemo" -v+
   ```

   The following table contains a description for each dbmlsync option used:

Option	Description
-c	Specifies the connection string.
-ot	Specifies the file to log messages to.
-e	Specifies the script version to synchronize with.
-v+	Specifies what information is logged. Using -v+ sets maximum verbose logging.

   An output screen appears indicating that the synchronization succeeded. SQL-based synchronization transferred rows in the client `T1` table to the `T1` table in the consolidated database.

2. Create a simulated client information file for use with the mlreplay utility.

   Create a new text file and write the following comma-separated list as displayed:

   ```
   mlreplay1,,1,
   mlreplay2,,2,
   mlreplay3,,3,
   mlreplay4,,4,
   mlreplay5,,5,
   ```

```
mlreplay6,,6,
mlreplay7,,7,
mlreplay8,,8,
mlreplay9,,9,
mlreplay10,,10,
```

3. Save the file as `mlreplay.csv` in your working directory.

   The client information file can be used to simulate ten remote clients.
4. Replay the recorded synchronization with simulated clients.

   Run the following command:

   ```
   mlreplay -ap -x tcpip -ot mlreplay.mls -sci mlreplay.csv
   recorded_protocol_mlreplay_svr_2.mlr
   ```

   The following table contains a description for each option used:

Option	Description
-ap	Adjusts the progress of synchronizations being replayed in a replay session so that the mlreplay utility does not cause progress offset mismatch warnings on the MobiLink server.
-x	Sets the protocol used to listen for synchronization requests.
-ot	Specifies the file to log messages.
-sci	Specifies the location of the client information file.

   The mlreplay utility stores information from the start of the connection to the end of the connection in a recorded protocol file named `recorded_protocol_mlreplay_svr_2.mlr`.
5. Open the `mlreplay.mls` log file with a text editor to review the outcome of the MobiLink replay.

Results

A synchronization is performed, the schema is cached on the MobiLink server, and a simulated client information file is created to replay the MobiLink protocol information about the simulated clients.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Simulating Multiple MobiLink Clients Using the MobiLink Replay Utility [page 319]

Previous task: Lesson 7: Restarting the MobiLink Server to Replay [page 331]

Next task: Lesson 9: Cleaning Up [page 335]

Related Information

MobiLink Replay Utility (mlreplay)

1.5.12.9 Lesson 9: Cleaning Up

Remove the tutorials materials from your computer.

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. Close any Interactive SQL, SQL Anywhere, MobiLink, and synchronization client windows by right-clicking each task bar item and choosing *Close*.
2. Delete all tutorial-related data sources:
   a. Start the ODBC Data Source Administrator.
   b. Click *Start* > *Programs* > *SQL Anywhere 17* > *Administration Tools* > *ODBC Data Source Administrator*.
   c. Select `cons` from the list of *User Data Sources*, and click *Remove*.
3. Delete to the directory containing your consolidated and remote databases.

Results

The tutorial materials are removed from your computer.

Task overview: Tutorial: Simulating Multiple MobiLink Clients Using the MobiLink Replay Utility [page 319]

Previous task: Lesson 8: Replaying Synchronization [page 333]

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