SYBASE iAnywhere

SQL Anywhere サーバ
データベース管理

改訂 2007 年 3 月

版権と商標

目次

はじめに

このマニュアルの内容

このマニュアルでは、SQL Anywhere データベースの実行、管理、設定について説明します。管理ユーティリティとオプションのほか、データベース接続、データベース・サーバ、データベース・ファイル、バックアップ・プロシージャ、セキュリティ、高可用性、Replication Server を使用したレプリケーションについて説明します。

対象読者

このマニュアルは、SQL Anywhere のすべてのユーザを対象としています。他のマニュアルと一緒に使用するように構成されています。

SQL Anywhere のマニュアル

このマニュアルは、SQL Anywhere のマニュアル・セットの一部です。この項では、マニュアル・セットに含まれる各マニュアルと使用法について説明します。

SQL Anywhere のマニュアル

SQL Anywhere の完全なマニュアルは、各マニュアルをまとめたオンライン形式とマニュアル別の PDF ファイルで提供されます。いずれの形式のマニュアルも、同じ情報が含まれ、次のマニュアルから構成されます。

- **『SQL Anywhere 10 - 紹介』** このマニュアルでは、データの管理および交換機能を提供する包括的なパッケージである SQL Anywhere 10 について説明します。SQL Anywhere を使用すると、サーバ環境、デスクトップ環境、モバイル環境、リモート・オフィス環境に適したデータベース・ベースのアプリケーションを迅速に開発できるようになります。
- **『SQL Anywhere 10 - 変更点とアップグレード』** このマニュアルでは、SQL Anywhere 10 とそれ以前のバージョンに含まれる新機能について説明します。
- **『SQL Anywhere サーバ - データベース管理』** このマニュアルでは、SQL Anywhere データベースの実行、管理、設定について説明します。管理ユーティリティとオプションのほか、データベース接続、データベース・サーバ、データベース・ファイル、バックアップ・プロシージャ、セキュリティ、高可用性、Replication Server を使用したレプリケーションについて説明します。
- **『SQL Anywhere サーバ - SQL の使用法』** このマニュアルでは、データベースの設計と作成の方法、データのインポート・エクスポート・変更の方法、データの検索方法、ストアド・プロシージャとトリガの構築方法について説明します。
- **『SQL Anywhere サーバ - SQL リファレンス』** このマニュアルは、SQL Anywhere で使用する SQL 言語の完全なリファレンスです。また、SQL Anywhere のシステム・ビューとシステム・プロシージャについても説明しています。
- **『SQL Anywhere サーバ - プログラミング』** このマニュアルでは、C、C++、Java プログラミング言語、Visual Studio .NET を使用してデータベース・アプリケーションを構築、配備する方法について説明します。Visual Basic や PowerBuilder などのツールのユーザは、それらのツールのプログラミング・インタフェースを使用できます。
- **『SQL Anywhere 10 - エラー・メッセージ』** このマニュアルでは、SQL Anywhere エラー・メッセージの完全なリストを、その診断情報とともに説明します。
- **『Mobile Link - クイック・スタート』** このマニュアルでは、セッションベースのリレーショナル・データベース同期システムである Mobile Link について説明します。Mobile Link テクノロジは、双方向レプリケーションを可能にし、モバイル・コンピューティング環境に非常に適しています。
- **『Mobile Link - サーバ管理』** このマニュアルでは、Mobile Link アプリケーションを設定して管理する方法について説明します。

- **『Mobile Link - クライアント管理』** このマニュアルでは、Mobile Link クライアントを設定、構成、同期する方法について説明します。Mobile Link クライアントには、SQL Anywhere または Ultra Light のいずれかのデータベースを使用できます。
- **『Mobile Link - サーバ起動同期』** このマニュアルでは、Mobile Link のサーバによって開始される同期について説明します。サーバによって開始される同期とは、統合データベースから同期またはその他のリモート・アクションの開始を可能にする Mobile Link の機能です。
- **『QAnywhere』** このマニュアルでは QAnywhere について説明します。QAnywhere は、従来のデスクトップ・クライアントやラップトップ・クライアント用のメッセージング・プラットフォームであるほか、モバイル・クライアントや無線クライアント用のメッセージング・プラットフォームでもあります。
- **『SQL Remote』** このマニュアルでは、モバイル・コンピューティング用の SQL Remote データ・レプリケーション・システムについて説明します。このシステムによって、SQL Anywhere の統合データベースと複数の SQL Anywhere リモート・データベースの間で、電子メールやファイル転送などの間接的リンクを使用したデータ共有が可能になります。
- **『SQL Anywhere 10 - コンテキスト別ヘルプ』** このマニュアルには、[接続] ダイアログ、クエリ・エディタ、Mobile Link モニタ、SQL Anywhere コンソール・ユーティリティ、インデックス・コンサルタント、Interactive SQL のコンテキスト別のヘルプが収録されています。
- **『Ultra Light - データベース管理とリファレンス』** このマニュアルでは、小型デバイス用 Ultra Light データベース・システムの概要を説明します。
- **『Ultra Light - AppForge プログラミング』** このマニュアルでは、Ultra Light for AppForge について説明します。Ultra Light for AppForge を使用すると、Palm OS、Symbian OS、または Windows CE を搭載しているハンドヘルド、モバイル、または埋め込みデバイスに対してデータベース・アプリケーションを開発、配備できます。
- **『Ultra Light - .NET プログラミング』** このマニュアルでは、Ultra Light.NET について説明します。Ultra Light.NET を使用すると、PC、ハンドヘルド、モバイル、埋め込みデバイスのデータベース・アプリケーションを開発し、これらのデバイスに配備できます。
- **『Ultra Light - M-Business Anywhere プログラミング』** このマニュアルは、Ultra Light for M-Business Anywhere について説明します。Ultra Light for M-Business Anywhere を使用すると、Palm OS、Windows CE、または Windows XP を搭載しているハンドヘルド、モバイル、または埋め込みデバイスに対して Web ベースのデータベース・アプリケーションを開発、配備できます。
- **『Ultra Light - C/C++ プログラミング』** このマニュアルでは、Ultra Light C および Ultra Light C++ のプログラミング・インタフェースについて説明します。Ultra Light を使用すると、ハンドヘルド、モバイル、埋め込みデバイスに対してデータベース・アプリケーションを開発、配備できます。

マニュアルの形式

SQL Anywhere のマニュアルは、次の形式で提供されています。

- **オンライン・マニュアル** オンライン・マニュアルには、SQL Anywhere の完全なマニュアルがあり、SQL Anywhere ツールに関する印刷マニュアルとコンテキスト別のヘルプの両方が含

まれています。オンライン・マニュアルは、製品のメンテナンス・リリースごとに更新されます。これは、最新の情報を含む最も完全なマニュアルです。

Windows オペレーティング・システムでオンライン・マニュアルにアクセスするには、[スタート] − [プログラム] − [SQL Anywhere 10] − [オンライン・マニュアル] を選択します。オンライン・マニュアルをナビゲートするには、左ウィンドウ枠で HTML ヘルプの目次、索引、検索機能を使用し、右ウィンドウ枠でリンク情報とメニューを使用します。

UNIX オペレーティング・システムでオンライン・マニュアルにアクセスするには、SQL Anywhere のインストール・ディレクトリまたはインストール CD に保存されている HTML マニュアルを参照してください。

- **PDF ファイル** SQL Anywhere の完全なマニュアル・セットは、Adobe Reader で表示できる Adobe Portable Document Format (pdf) 形式のファイルとして提供されています。

  Windows では、PDF 形式のマニュアルはオンライン・マニュアルの各ページ上部にある PDF のリンクから、または Windows の [スタート] メニュー ([スタート] − [プログラム] − [SQL Anywhere 10] − [オンライン・マニュアル - PDF フォーマット]) からアクセスできます。

  UNIX では、PDF 形式のマニュアルはインストール CD にあります。

表記の規則

この項では、このマニュアルで使用されている書体およびグラフィック表現の規則について説明します。

SQL 構文の表記規則

SQL 構文の表記には、次の規則が適用されます。

- **キーワード**　SQL キーワードはすべて次の例に示す ALTER TABLE のように大文字で表記します。

  ALTER TABLE [*owner.*]*table-name*

- **プレースホルダ**　適切な識別子または式で置き換えられる項目は、次の例に示す *owner* や *table-name* のように表記します。

  ALTER TABLE [*owner.*]*table-name*

- **繰り返し項目**　繰り返し項目のリストは、次の例に示す *column-constraint* のように、リストの要素の後ろに省略記号 (ピリオド 3 つ …) を付けて表します。

  ADD *column-definition* [*column-constraint*, …]

  複数の要素を指定できます。複数の要素を指定する場合は、各要素間をカンマで区切る必要があります。

- **オプション部分**　文のオプション部分は角カッコで囲みます。

  RELEASE SAVEPOINT [*savepoint-name*]

  この例では、角カッコで囲まれた *savepoint-name* がオプション部分です。角カッコは入力しないでください。

- **オプション**　項目リストから 1 つだけ選択する場合や、何も選択しなくてもよい場合は、項目間を縦線で区切り、リスト全体を角カッコで囲みます。

  [**ASC** | **DESC**]

  この例では、ASC と DESC のどちらか 1 つを選択しても、選択しなくてもかまいません。角カッコは入力しないでください。

- **選択肢**　オプションの中の 1 つを必ず選択しなければならない場合は、選択肢を中カッコで囲み、縦棒で区切ります。

  [**QUOTES** { **ON** | **OFF** }]

  QUOTES オプションを使用する場合は、ON または OFF のどちらかを選択する必要があります。角カッコと中カッコは入力しないでください。

オペレーティング・システムの表記規則

- **Windows**　デスクトップおよびラップトップ・コンピュータ用の Microsoft Windows オペレーティング・システムのファミリのことです。Windows ファミリには Windows Vista や Windows XP も含まれます。
- **Windows CE**　Microsoft Windows CE モジュラ・オペレーティング・システムに基づいて構築されたプラットフォームです。Windows Mobile や Windows Embedded CE などのプラットフォームが含まれます。

  Windows Mobile は Windows CE 上に構築されています。これにより、Windows のユーザ・インタフェースや、Word や Excel といったアプリケーションの小規模バージョンなどの追加機能が実現されています。Windows Mobile は、モバイル・デバイスで最も広く使用されています。

  SQL Anywhere の制限事項や相違点は、基盤となっているオペレーティング・システム (Windows CE) に由来しており、使用しているプラットフォーム (Windows Mobile など) に依存していることはほとんどありません。
- **UNIX**　特に記述がないかぎり、UNIX は Linux プラットフォームと UNIX プラットフォームの両方のことです。

ファイルの命名規則

マニュアルでは、パス名やファイル名などのオペレーティング・システムに依存するタスクと機能を表すときは、通常 Windows の表記規則が使用されます。ほとんどの場合、他のオペレーティング・システムで使用される構文に簡単に変換できます。

- **ディレクトリ名とパス名**　マニュアルでは、ドライブを示すコロンや、ディレクトリの区切り文字として使用する円記号など、Windows の表記規則を使用して、ディレクトリ・パスのリストを示します。次に例を示します。

  ```
  MobiLink¥redirector
  ```

  UNIX、Linux、Mac OS X では、代わりにスラッシュを使用してください。次に例を示します。

  ```
  MobiLink/redirector
  ```

  SQL Anywhere がマルチプラットフォーム環境で使用されている場合、プラットフォーム間でのパス名の違いに注意する必要があります。
- **実行ファイル**　マニュアルでは、実行ファイルの名前は、Windows の表記規則が使用され、拡張子 *.exe* が付きます。UNIX、Linux、Mac OS X では、実行ファイルの名前には拡張子は付きません。NetWare では、実行ファイルの名前には、拡張子 *.nlm* が付きます。

  たとえば、Windows では、ネットワーク・データベース・サーバは *dbsrv10.exe* です。UNIX、Linux、Mac OS X では、*dbsrv10* になります。NetWare では、*dbsrv10.nlm* になります。
- **install-dir**　インストール・プロセスでは、SQL Anywhere をインストールするロケーションを選択できます。マニュアルでは、このロケーションは *install-dir* という表記で示されます。

インストールが完了すると、環境変数 SQLANY10 によって SQL Anywhere コンポーネントがあるインストール・ディレクトリのロケーション (*install-dir*) が指定されます。SQLANYSH10 は、SQL Anywhere が他の Sybase アプリケーションと共有しているコンポーネントがあるディレクトリのロケーションを指定します。

オペレーティング・システム別の *install-dir* のデフォルト・ロケーションの詳細については、「SQLANY10 環境変数」 315 ページを参照してください。

- **samples-dir** インストール・プロセスでは、SQL Anywhere に含まれるサンプルをインストールするロケーションを選択できます。マニュアルでは、このロケーションは *samples-dir* という表記で示されます。

  インストールが完了すると、環境変数 SQLANYSAMP10 によってサンプルがあるディレクトリのロケーション (*samples-dir*) が指定されます。Windows の [スタート] メニューから、[プログラム] – [SQL Anywhere 10] – [サンプル・アプリケーションおよびプロジェクト] を選択すると、このディレクトリで [Windows エクスプローラ] ウィンドウが表示されます。

  オペレーティング・システム別の *samples-dir* のデフォルト・ロケーションの詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

- **環境変数** マニュアルでは、環境変数設定が引用されます。Windows では、環境変数を参照するのに、構文 *%envvar%* が使用されます。UNIX、Linux、Mac OS X では、環境変数を参照するのに、構文 *$envvar* または *${envvar}* が使用されます。

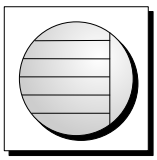
  UNIX、Linux、Mac OS X 環境変数は、*.cshrc* や *.tcshrc* などのシェルとログイン・スタートアップ・ファイルに格納されます。

グラフィック・アイコン

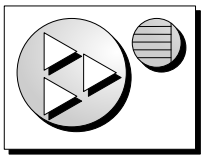
このマニュアルでは、次のアイコンを使用します。

- クライアント・アプリケーション

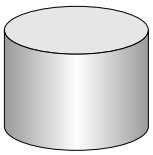
- SQL Anywhere などのデータベース・サーバ



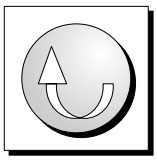
- Ultra Light アプリケーション



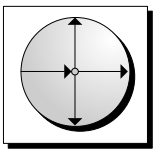
- データベース。高度な図では、データベースとデータベースを管理するデータ・サーバの両方をこのアイコンで表します。



- レプリケーションまたは同期のミドルウェア。ソフトウェアのこれらの部分は、データベース間のデータ共有を支援します。たとえば、Mobile Link サーバ、SQL Remote Message Agent などが挙げられます。



- Sybase Replication Server



- プログラミング・インタフェース

詳細情報の検索／フィードバックの提供

詳細情報の検索

詳しい情報やリソース (コード交換など) については、iAnywhere Developer Network (http://www.ianywhere.com/developer/) を参照してください。

ご質問がある場合や支援が必要な場合は、次に示す Sybase iAnywhere ニュースグループのいずれかにメッセージをお寄せください。

ニュースグループにメッセージをお送りいただく際には、ご使用の SQL Anywhere バージョンのビルド番号を明記し、現在発生している問題について詳しくお知らせくださいますようお願いいたします。バージョン情報は、コマンド・プロンプトで **dbeng10 -v** と入力して確認できます。

ニュースグループは、ニュース・サーバ *forums.sybase.com* にあります (ニュースグループにおけるサービスは英語でのみの提供となります)。以下のニュースグループがあります。

- sybase.public.sqlanywhere.general
- sybase.public.sqlanywhere.linux
- sybase.public.sqlanywhere.mobilink
- sybase.public.sqlanywhere.product_futures_discussion
- sybase.public.sqlanywhere.replication
- sybase.public.sqlanywhere.ultralite
- ianywhere.public.sqlanywhere.qanywhere

ニュースグループに関するお断り
iAnywhere Solutions は、ニュースグループ上に解決策、情報、または意見を提供する義務を負うものではありません。また、システム・オペレータ以外のスタッフにこのサービスを監視させて、操作状況や可用性を保証する義務もありません。
iAnywhere のテクニカル・アドバイザとその他のスタッフは、時間のある場合にかぎりニュースグループでの支援を行います。こうした支援は基本的にボランティアで行われるため、解決策や情報を定期的に提供できるとはかぎりません。支援できるかどうかは、スタッフの仕事量に左右されます。

フィードバック

このマニュアルに関するご意見、ご提案、フィードバックをお寄せください。

マニュアルに関するご意見、ご提案は、SQL Anywhere ドキュメンテーション・チームの iasdoc@ianywhere.com 宛てに電子メールでお寄せください。このアドレスに送信された電子メールに返信はいたしませんが、お寄せいただいたご意見、ご提案は必ず読ませていただきます。

マニュアルまたはソフトウェアについてのフィードバックは、上記のニュースグループを通してお寄せいただいてもかまいません。

パート I. データベースの開始とデータベースへの接続

パート I では、SQL Anywhere データベース・サーバの起動方法と、クライアント・アプリケーションからデータベースに接続する方法について説明します。

第 1 章

チュートリアル：サンプル・データベースの使用

目次

レッスン 1 ：サンプル・データベースのコピーの作成

このチュートリアルでは、サンプル・データベースに重点を置きます。サンプル・データベースは、限られた種類のスポーツ衣料品を製造する小企業を想定して作られています。データベースには、この企業の内部情報 (従業員、部署、財務データ) とともに、製品情報 (製品) や販売情報 (受注、顧客、連絡先) が入っています。サンプル・データベースに入っている情報は、すべて架空のものです。

「サンプル・データベースについて」 『SQL Anywhere 10 - 紹介』を参照してください。

始める前に、サンプル・データベースのコピーを作成し、変更後にリストアできるようにしておきます。

♦ **サンプル・データベースのコピーを作成するには、次の手順に従います。**

1. このチュートリアルで使用するサンプル・データベースのコピーを保存するディレクトリ (*c:¥demodb* など) を作成します。
2. サンプル・データベースを *samples-dir¥demo.db* から *c:¥demodb* にコピーします。

   samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

レッスン 2：SQL Anywhere データベース・サーバの起動

この項では、サンプル・データベースを実行する SQL Anywhere パーソナル・データベース・サーバを起動する方法について説明します。

ネットワーク・データベース・サーバの起動の詳細については、「ネットワーク上のサーバへの接続」 70 ページを参照してください。

♦ サンプル・データベースを実行するパーソナル・データベース・サーバを起動するには、次の手順に従います (Windows の場合)。

- [スタート] メニューから、[プログラム] – [SQL Anywhere 10] – [SQL Anywhere] – [パーソナル・サーバのサンプル] を選択します。

  これによって、サンプル・データベースを実行するパーソナル・データベース・サーバが起動します。サーバ・ウィンドウが表示されて、すぐに閉じます。データベース・サーバは、システム・トレイにアイコンとして表示されます。

♦ サンプル・データベースを実行するパーソナル・データベース・サーバを起動するには、次の手順に従います (コマンド・プロンプトの場合)。

1. コマンド・プロンプトを開きます。
2. SQL Anywhere インストール・ディレクトリに移動します。
3. サンプル・データベースを実行するパーソナル・データベース・サーバを起動します。

   次のコマンドを入力して、パーソナル・データベース・サーバを起動し、サーバ名 demo10 を -n サーバ・オプションを使用して割り当て、サンプル・データベースに接続します。

   ```
   dbeng10 -n demo10 C:¥Documents and Settings¥All Users¥Documents¥SQL Anywhere 10
   ¥Samples¥demo.db
   ```

   Windows の場合、データベース・サーバはシステム・トレイにアイコンとして表示されます。

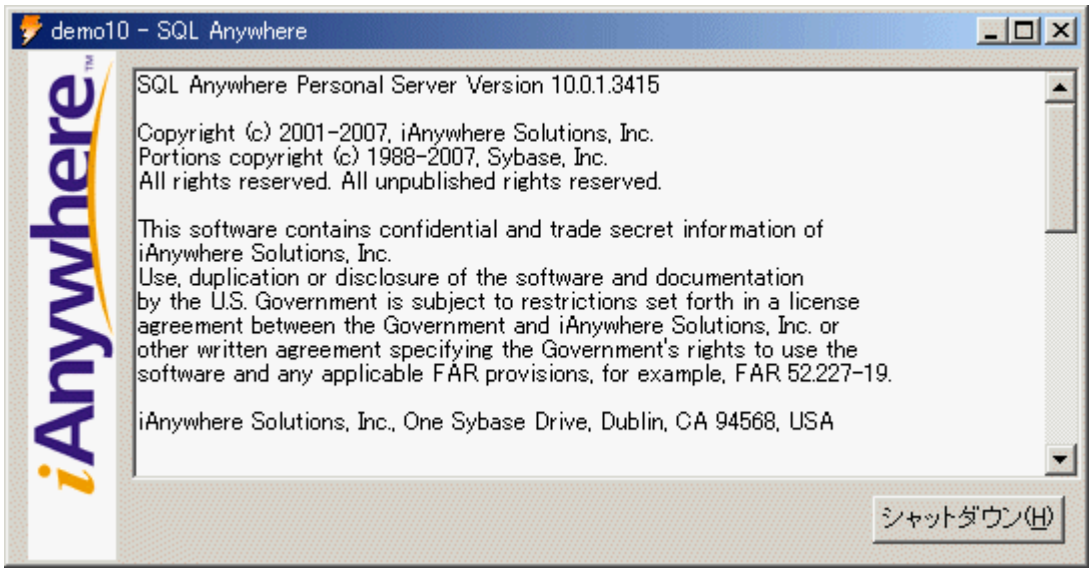
参照

- 「データベース・サーバの実行」 11 ページ
- 「データベース・サーバ」 137 ページ

レッスン 3：サーバ・メッセージ・ウィンドウの表示

サンプル・データベースを実行するパーソナル・データベース・サーバが正常に起動されました。ただし、まだデータベース内のデータを表示したり操作したりはできません。

SQL Anywhere のアイコンは単に表示されているだけです。システム・トレイに表示されている SQL Anywhere パーソナル・サーバのアイコンをダブルクリックすると、Windows 画面にサーバ・メッセージ・ウィンドウが表示されます。



サーバ・メッセージ・ウィンドウには次の情報が表示されます。

- **サーバ名**　タイトル・バーに表示されている名前 (この例では、**demo10**) が「**サーバ名**」です。このチュートリアルでは、サーバ名を -n サーバ・オプションを使用して割り当てています。サーバ名を指定しない場合は、最初に起動されたデータベースの名前になります。この名前は、アプリケーションがデータベースに接続するときに使用します。「サーバとデータベースの命名」 19 ページを参照してください。
- **バージョンとビルド番号**　サーバ名に続く数字 (**10.0.0.2435** など) は、「バージョン番号とビルド番号」です。バージョン番号は SQL Anywhere の特定のリリースを表し、ビルド番号はソフトウェアがコンパイルされた特定のインスタンスに関連します。
- **起動情報**　データベース・サーバは、起動時に、データベース要求を処理するときに使用するメモリを別に設定します。これを「**キャッシュ**」と呼びます。キャッシュ・メモリの量は、このウィンドウに表示されます。キャッシュは固定サイズの「**ページ**」で構成されていますが、このページのサイズもウィンドウ内に表示されます。
- **データベース情報**　データベース・ファイルの名前とそのトランザクション・ログ・ファイルがウィンドウに表示されます。

ここでは、起動時のキャッシュ・サイズとページのサイズはデフォルト値です。このチュートリアルの場合も含めて、多くの場合、デフォルトの起動オプションが適しています。

レッスン 4：データベース・サーバの停止

起動したデータベース・サーバを停止できます。

Windows では、サーバ・メッセージ・ウィンドウで [シャットダウン] をクリックしてデータベース・サーバを停止できます。

♦ サンプル・データベースを実行しているデータベース・サーバを停止するには、次の手順に従います (Windows の場合)。

1. Windows タスクバーにある SQL Anywhere のアイコンをダブルクリックします。

   サーバ・メッセージ・ウィンドウが表示されます。

2. [シャットダウン] をクリックします。

♦ サンプル・データベースを実行するデータベース・サーバを停止するには、次の手順に従います (コマンド・プロンプトの場合)。

1. コマンド・プロンプトを開きます。
2. SQL Anywhere インストール・ディレクトリに移動します。
3. 次のコマンドを実行して、サンプル・データベースを実行するパーソナル・データベース・サーバを停止します。

   ```
   dbstop demo10
   ```

   サーバ停止ユーティリティ (dbstop) はコマンド・プロンプトでのみ実行できます。NetWare では使用できません。「サーバ停止ユーティリティ (dbstop)」 732 ページを参照してください。
4. *c:¥demodb* ディレクトリを削除します。

まとめ

このチュートリアルでは、サンプル・データベースのコピーの作成方法、サンプル・データベースを実行するデータベース・サーバの起動方法、サーバ・メッセージ・ウィンドウの表示方法について学習しました。また、データベース・サーバの停止方法についても学習しました。

参照

- 「Interactive SQL の起動」 586 ページ
- 「Sybase Central、Interactive SQL、SQL Anywhere コンソール・ユーティリティからの接続」 64 ページ

第 2 章

データベース・サーバの実行

目次

SQL Anywhere データベース・サーバ実行の概要

SQL Anywhere では、2 つのバージョンのデータベース・サーバを提供しています。

- **パーソナル・データベース・サーバ**　この実行プログラムは、ネットワークを介したクライアント／サーバ通信をサポートしていません。パーソナル・データベース・サーバは、シングルユーザによる同一コンピュータ上での (たとえば組み込みデータベース・サーバとしての) 使用を目的としていますが、開発作業にも適してします。

  Windows CE 以外の Windows オペレーティング・システムでは、パーソナル・サーバの実行プログラムの名前は *dbeng10.exe* です。UNIX オペレーティング・システムでは、dbeng10 です。Windows CE および NetWare でサポートされているのはネットワーク・サーバのみです。

- **ネットワーク・データベース・サーバ**　複数ユーザでの使用を目的としたこの実行プログラムは、ネットワーク経由のクライアント／サーバ通信をサポートします。

  Windows CE を含む Windows オペレーティング・システムでは、ネットワーク・サーバ実行プログラムの名前は *dbsrv10.exe* です。Novell NetWare では *dbsrv10.nlm*、Linux および UNIX オペレーティング・システムでは dbsrv10 です。

サーバ間の相違点

パーソナル・サーバとネットワーク・サーバの 2 つのサーバの要求処理エンジンはまったく同じです。各サーバで、まったく同じ SQL とデータベース機能をサポートします。主な相違点には次のようなものがあります。

- **ネットワーク・プロトコルのサポート**　ネットワークを介した通信をサポートするのはネットワーク・サーバのみです。
- **接続数**　パーソナル・サーバには、同時接続数は 10 という制限があります。ネットワーク・サーバの最大接続数は、ライセンスによって異なります。「サーバ・ライセンス取得ユーティリティ (dblic)」 701 ページを参照してください。
- **CPU 数**　per-seat ライセンスの場合、ネットワーク・データベース・サーバはコンピュータで使用可能なすべての CPU を使用します (デフォルト)。CPU ベースのライセンスの場合、ネットワーク・データベース・サーバはライセンスを受けたプロセッサ数のみ使用可能です。また、パーソナル・データベース・サーバとランタイム・データベース・サーバは、いずれも 1 つのプロセッサしか使用できません。
- **起動時のデフォルト**　パーソナル・サーバとマルチユーザ用のネットワーク・サーバとでは、その用途を反映して起動時のデフォルトが若干異なります。

必要なネットワーク・ソフトウェア

SQL Anywhere ネットワーク・サーバを実行している場合、適切なネットワーク・ソフトウェアがインストールされて実行されている必要があります。

SQL Anywhere ネットワーク・サーバは、Windows、NetWare、Linux、UNIX オペレーティング・システムで使用できます。

SQL Anywhere では TCP/IP ネットワーク プロトコルもサポートされています。Novell NetWare 用に SPX プロトコルもサポートされています。

第一段階

単一のデータベースを実行するパーソナル・サーバを起動する方法は複数あります。

- Windows の場合、[スタート] メニューから、[プログラム] – [SQL Anywhere 10] – [SQL Anywhere] – [パーソナル・サーバのサンプル] を選択する。
- *demo.db* があるディレクトリで次のコマンドを実行して、パーソナル・サーバと *demo.db* という名前のデータベース・サーバを起動する。

  ```
  dbeng10 demo
  ```

- 接続文字列でデータベース・ファイル名を使用する。

  「組み込みデータベースへの接続」 68 ページを参照してください。

コマンドを指定する場所

コマンドは、使用しているオペレーティング・システムに応じて、複数の方法で指定できます。次に例を示します。

- システムのコマンド・プロンプトでコマンドを入力する。
- コマンドをショートカットまたはデスクトップ・アイコンに配置する。
- バッチ・ファイルでコマンドを実行する。
- コマンドを StartLine (START) 接続パラメータとして文字列に含める。

  「StartLine 接続パラメータ [START]」 262 ページを参照してください。

基本的なコマンドの指定方法は、プラットフォームによって若干違いがあります。それらについて、次の項で説明します。

データベース・サーバの起動

データベースの開始方法は、使用するオペレーティング・システムによって若干違いがあります。この項では、サポートされている各オペレーティング・システムで、デフォルトの設定値を使用して単一のデータベースを実行する場合のコマンドの入力方法について説明します。

注意

- 特に指定のないかぎり、これらのコマンドは、パーソナル・サーバ (**dbeng10**) を起動します。ネットワーク・サーバを起動する場合は、**dbeng10** を **dbsrv10** に置き換えてください。
- データベース・ファイルがコマンドを開始するディレクトリに含まれている場合は、*path* を指定する必要はありません。

- *database-file* にファイルの拡張子を指定しないと、拡張子は *.db* であると見なされます。

♦ デフォルト・オプションを使用してパーソナル・データベース・サーバを起動するには、次の手順に従います (Windows CE 以外の Windows の場合)。

1. コマンド・プロンプトを開きます。
2. 次のコマンドを入力します。

   ```
   dbeng10 path¥database-file
   ```

   データベース・ファイルを省略した場合は、[参照] を使用するとデータベース・ファイルを検索できる [サーバ起動オプション] ダイアログが表示されます。

Windows CE でのデータベース・サーバの起動については、「Windows CE デバイスで実行中のデータベースへの接続」 1039 ページを参照してください。

♦ デフォルト・オプションを使用してパーソナル・データベース・サーバを起動するには、次の手順に従います (UNIX の場合)。

1. コマンド・プロンプトを開きます。
2. 次のコマンドを入力します。

   ```
   dbeng10 path/database-file
   ```

Novell NetWare 用のパーソナル・サーバはありません。ネットワーク・サーバだけが用意されています。次のコマンドにより、NetWare でネットワーク・サーバが起動されます。

♦ デフォルト・オプションを使用してデータベース・サーバを起動するには、次の手順に従います (NetWare の場合)。

- NetWare 用のデータベース・サーバは NetWare Loadable Module (NLM) (*dbsrv10.nlm*) と呼ばれます。NLM は、NetWare サーバで実行できるプログラムです。データベース・サーバは、次のコマンドで NetWare サーバにロードします。

  ```
  load dbsrv10.nlm path¥database-file
  ```

  データベース・ファイルは NetWare ボリュームに入れてください。一般的なファイル名は *DB:¥database¥sales.db* の形式です。

  Novell リモート・コンソール・ユーティリティを使用して、クライアント・コンピュータからサーバをロードできます。詳細については、NetWare のマニュアルを参照してください。

  NetWare *autoexec.ncf* ファイルにコマンドを指定しておくと、NetWare サーバを起動するたびに、SQL Anywhere が自動的にロードされます。

補足事項

パーソナル・サーバは前述のように簡単に起動できますが、実際の運用環境でデータベース・サーバを起動する場合には、他にもさまざまな側面があります。次に例を示します。

- 各種の「**オプション**」を選択して、キャッシュに使用するメモリ量、使用する CPU の数 (ネットワーク・データベース・サーバを実行中のマルチプロセッサ・コンピュータ上)、使用するネットワーク・プロトコル (ネットワーク・サーバのみ) などを指定できます。オプションは、SQL Anywhere の動作とパフォーマンスをチューニングする主要な方法の 1 つです。
- サーバは、Windows の「**サービス**」として実行できます。これによって、ユーザがログオフしてもサーバの実行を継続できます。
- パーソナル・サーバは、アプリケーションから起動し、アプリケーションの終了時に同時にシャットダウンできます。これは、データベース・サーバを「**組み込みデータベース**」として使用するときに一般的な方法です。

これらのオプションについては、この後詳しく説明します。

データベース・サーバの起動

サーバ・コマンドは、通常次のような形式になっています。

executable [*server-options*] [*database-file* [*database-options*], …]

オプションもデータベース・ファイルも指定しなかった場合、Windows オペレーティング・システムではダイアログ・ボックスが表示され、[参照] を使用してデータベース・ファイルを検索できます。

データベース・サーバのコマンドの要素には、次のようなものがあります。

- **実行プログラム**　これは、パーソナル・サーバまたはネットワーク・サーバのいずれかを指定できます。

  各オペレーティング・システムでの実行プログラム名の詳細については、「SQL Anywhere データベース・サーバ実行の概要」 12 ページを参照してください。

  この章では、ネットワーク特有のオプションについて説明する場合を除き、サンプル・コマンドにはパーソナル・サーバを使用します。ネットワーク・サーバのオプションはパーソナル・サーバのものと非常に良く似ています。

- **サーバ・オプション**　これらのオプションは、実行中のすべてのデータベースに対するデータベース・サーバの動作を制御します。

- **データベース・ファイル**　1 つまたは複数のデータベース・ファイル名を指定するか、まったく指定しないこともできます。指定された各データベースが起動され、引き続きアプリケーションで使用できます。

  警告
  データベース・ファイルとトランザクション・ログ・ファイルは、データベース・サーバと同じ物理コンピュータに保存してください。または SAN や iSCSI 設定でアクセスできるようにしてください。リモート・ネットワーク・ディレクトリにデータベース・ファイルやトランザクション・ログ・ファイルを配置すると、パフォーマンスが低下したり、データが破壊されたり、サーバが不安定になったりする可能性があります。
  詳細については、http://www.ianywhere.com/developer/technotes/asa_db_file_stored_remotely.html を参照してください。
  最適な結果を得るために、トランザクション・ログは、データベース・ファイルとは別のディスクに保存してください。「トランザクション・ログ」 816 ページを参照してください。

- **データベース・オプション**　開始するデータベース・ファイルごとに、その動作の特定の面を制御するデータベース・オプションを指定できます。

  「SQL Anywhere データベース・サーバ」 138 ページを参照してください。

この章では、複数のオプションがある例については、別々の行で示してわかりやすくしています。設定ファイルには、サンプルと同じ形で記述できます。これらのオプションをコマンド・プロンプトに直接入力する場合は、すべてを 1 行に入力します。

大文字と小文字の区別

データベース・オプションとサーバ・オプションでは、通常は大文字と小文字が区別されます。オプションはすべて小文字で入力してください。

使用可能なオプションのリスト表示

- **データベース・サーバのオプションをリスト表示するには、次の手順に従います。**

1. コマンド・プロンプトを開きます。
2. 次のコマンドを入力します。

```
dbeng10 -?
```

データベース・サーバの動作のロギング

サーバの動作をロギングすると、開発プロセスとトラブルシューティングのときに特に役立ちます。

ファイルへのロギング結果の出力

ロギング結果は、サーバ・メッセージ・ウィンドウに送信されます。また、-o オプションを使用して結果をログ・ファイルにも送信できます。次のコマンドでは、結果を *dbsrv.log* という名前のログ・ファイルに送ります。

```
dbsrv10 -o dbsrv.log -c ...
```

コンソール・ログ・ファイルのサイズを制御したり、ファイルが最大サイズに達したときの処理を指定したりできます。

- -o オプションを使用して、ログ・ファイルを使用することとファイル名を指定します。
- -ot オプションを使用して、ログ・ファイルを使用することとファイル名を指定すると、メッセージが送信される前にログ・ファイルの前の内容が削除されます。
- -o または -ot の他に -on オプションを使用してサイズを指定すると、そのサイズに達したときに、拡張子 .old を付けてログ・ファイルの名前が変更され、元の名前の新しいファイルが使用されます。
- -o または -ot の他に -os オプションを使用してサイズを指定すると、そのサイズに達したときに、日付と連番に基づいた新しい名前の新しいログ・ファイルが使用されます。

起動エラー、致命的なエラー、アサーションのロギング先をそれぞれ別ファイルにするには、-oe オプションを指定します。

参照

- 「-o サーバ・オプション」 182 ページ

一般的なオプション

この項では、最も一般的なオプションのいくつかとそれらを使用する状況について説明します。次のものがあります。

- 設定ファイルの使用
- サーバとデータベースの命名
- パフォーマンス
- パーミッション
- 最大ページ・サイズ
- 特殊モード
- スレッド
- ネットワーク通信 (ネットワーク・サーバのみ)

設定ファイルを使用したサーバ起動オプションの保存

オプションの拡張セットを使用する場合は、それらを設定ファイルに保存し、サーバ・コマンドでそのファイルを呼び出すことができます。設定ファイルには、複数行にわたってオプションを保存できます。たとえば、次の設定ファイルは、パーソナル・データベース・サーバとサンプル・データベースを起動します。また、キャッシュを 10 MB に設定し、パーソナル・サーバのこのインスタンスの名前を **Elora** にします。最初の文字に # が使用されている行は、コメントとして処理されます。

```
# Configuration file for server Elora
-n Elora
-c 10M
samples-dir¥demo.db
```

この例では、*samples-dir* は SQL Anywhere サンプル・ディレクトリの名前です。UNIX のファイル・パスには、円記号の代わりに、通常のスラッシュを使用します。

samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

ファイルに *sample.cfg* という名前を付けた場合は、これらのオプションを次のように使用できます。

```
dbeng10 @sample.cfg
```

参照

- 「@data サーバ・オプション」 147 ページ
- 「設定ファイルの使用」 637 ページ

サーバとデータベースの命名

-n は、サーバ・オプション (サーバ名の指定) またはデータベース・オプション (データベース名の指定) として使用できます。

サーバ名とデータベース名は、データベースに接続するときにクライアント・アプリケーションが使用する接続パラメータに含まれます。サーバ名は、タスクトレイ・アイコンとサーバ・メッセージ・ウィンドウのタイトル・バーに表示されます。

サーバの命名

データベース・サーバに名前を付けると、ネットワーク上の他のサーバ名との重複を防ぐことができます。また、クライアント・アプリケーションのユーザにわかりやすい名前を提供できます。サーバは、停止するまでその名前を維持します。サーバ名を指定しない場合は、最初に起動されたデータベースの名前になります。

最初のデータベース・ファイルの前に -n オプションを指定すると、サーバに名前を付けることができます。たとえば、次のコマンドは、サンプル・データベースでサーバを起動し、そのサーバに Cambridge という名前を付けます。

```
dbeng10 -n Cambridge samples-dir¥demo.db
```

サーバ名を指定すると、データベースを起動せずにデータベース・サーバを起動できます。次のコマンドは、データベースを起動せずに Galt という名前のサーバを起動します。

```
dbeng10 -n Galt
```

サーバ名の長さの上限は、使用するプロトコルによって異なります。

プロトコル	トランケーションの長さ
TCP/IP	250 バイト
共有メモリ	250 バイト
SPX	32 バイト

実行中のサーバでのデータベースの起動については、「データベースの開始と停止」 34 ページを参照してください。

注意
Windows と UNIX では、データベース・サーバがバージョン 10.0.0 以降で、名前が次の長さを超えている場合、バージョン 9.0.2 以前のクライアントから接続することはできません。

- Windows 共有メモリの場合は、40 バイト
- UNIX 共有メモリの場合は、31 バイト
- TCP/IP の場合は、40 バイト

データベースの命名

クライアント・アプリケーションのユーザにわかりやすいデータベースの名前を提供できます。データベースは、停止されるまでその名前で識別されます。

データベース名を指定しない場合、デフォルト名はデータベース・ファイル名のルートとなります (*.db* 拡張子を持たないファイル名)。たとえば、次のコマンドでは、最初のデータベース名は mydata、次のデータベース名は mysales です。

```
dbeng10 c:¥mydata.db c:¥sales¥mysales.db
```

データベース・ファイルの後に -n オプションを指定すると、データベースに名前を付けることができます。たとえば次のコマンドでは、サンプル・データベースを起動し、それに MyDB という名前を付けます。

```
dbeng10 samples-dir¥demo.db -n MyDB
```

大文字と小文字の区別

サーバ名とデータベース名は、文字セットがシングルバイトの場合は大文字と小文字が区別されません。「接続文字列と文字セット」 344 ページを参照してください。

コマンド・ラインからパフォーマンスとメモリを制御する

データベース・サーバのパフォーマンスに大きく影響するオプションには、次のようなものがあります。

- **キャッシュ・サイズ** -c オプションは、SQL Anywhere がキャッシュとして使用するメモリ容量を制御します。このオプションは、パフォーマンスに大きく影響します。

  一般的に、データベース・サーバが利用できるメモリが多いほど、実行速度が速くなります。キャッシュには何度も要求される情報が保持されます。ディスクの情報にアクセスするよりも、キャッシュ内の情報にアクセスする方がはるかに速くなります。デフォルトの初期キャッシュ・サイズは、物理メモリの容量、オペレーティング・システムとデータベース・ファイルのサイズに基づいて計算されます。Windows オペレーティング・システムと UNIX オペレーティング・システムでは、利用可能キャッシュを使い切ると、データベース・サーバは自動的にキャッシュを拡大します。

  サーバ・メッセージ・ウィンドウには起動時のキャッシュ・サイズが表示されます。また、次の文を使用して現在のキャッシュ・サイズを取得することもできます。

  ```
  SELECT PROPERTY( 'CacheSize' )
  ```

  パフォーマンス・チューニングの詳細については、「パフォーマンスのモニタリングと改善」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

  キャッシュ・サイズ制御の詳細については、「-c サーバ・オプション」 149 ページを参照してください。

- **マルチプログラミング・レベル** データベース・サーバのマルチプログラミング・レベルは、同時に実行できるサーバ・タスクの最大数を指定するものです。一般に、マルチプログラミング・レベルが高いほど、同時に実行する要求の数が増えるため、サーバの全体的なスループットは向上します。ただし、マルチプログラミング・レベルを高く設定すると、同じリソースを使用する要求が複数存在するときに、余分な競合が発生し、トランザクションの応答時間が長くなります。場合によっては、システムのスループットが一層低下することもあります。サーバのマルチプログラミング・レベルを設定するには、-gn オプションを使用します。「-gn サーバ・オプション」 172 ページと「データベース・サーバのマルチプログラミング・レベルの設定」 27 ページを参照してください。

- **プロセッサの数** ネットワーク・データベース・サーバを使用するマルチプロセッサ・コンピュータを実行している場合、-gt オプションを使用してプロセッサ数を設定できます。「-gt サーバ・オプション」 174 ページと「SQL Anywhere でのスレッド」 23 ページを参照してください。
- **その他のパフォーマンスに関連するオプション** ネットワークのパフォーマンスをチューニングするオプションには、-gb (データベース処理優先度) と -u (バッファ・ディスク I/O) などいくつかのオプションがあります。

  「SQL Anywhere データベース・サーバ」 138 ページを参照してください。

コマンド・ラインからパーミッションを制御する

ある特定のグローバル・オペレーションの実行に必要なパーミッションを制御するオプションがあります。制御されるパーミッションには、データベースの開始と停止、データのロードとアンロード、データベース・ファイルの作成と削除などを行うものが含まれます。「安全な方法でのデータベース・サーバの実行」 934 ページを参照してください。

最大ページ・サイズの設定

データベース・サーバ・キャッシュは、固定サイズのメモリ領域である「**ページ**」に配置されます。サーバは停止するまで 1 つのキャッシュを使用するので、ページはすべて同じサイズでなければなりません。

データベース・ファイルも、コマンド・ラインで指定されたサイズのページに配列されます。どのデータベース・ページも、キャッシュ・ページに適合していなければなりません。デフォルトでは、サーバ・ページ・サイズは、コマンド・ラインで指定されたデータベースの最大ページ・サイズと同じ大きさです。サーバがいったん起動すると、サーバ・ページより大きいページ・サイズのデータベースを起動することはできません。

サーバの起動後に大きなページ・サイズを持つデータベースを起動するには、-gp オプションでページ・サイズを指定してサーバを起動します。より大きいページ・サイズを使用する場合は、必ずキャッシュ・サイズを増やしてください。キャッシュ・サイズを変更しないと、大きなページの一部だけが保管され、領域調整の柔軟性が低くなります。

次のコマンドで、64 MB のキャッシュを予約し、最大 8192 バイトのページ・サイズを使用するデータベースを収容できるサーバを起動します。

```
dbsrv10 -gp 8192 -c 64M -n myserver
```

特殊モードでの実行

特定の目的のために、SQL Anywhere を特殊モードで実行できます。

- **読み込み専用** -r オプションを入力すると、データベースを読み込み専用モードで起動できます。読み込み専用モードで監査が有効になっている場合は、データベースを起動できません。「-r サーバ・オプション」 189 ページを参照してください。

- **バルク・ロード**　これは、Interactive SQL の INPUT コマンドを使用してデータベースに大量のデータをロードするときに便利です。LOAD TABLE を使用してデータをバルク・ロードする場合は、-b オプションは使用しないでください。

  「-b サーバ・オプション」 148 ページと「データのインポートとエクスポート」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **トランザクション・ログなしの起動**　-f データベース・オプションは、リカバリ時、つまりトランザクション・ログの消失後にデータベース・サーバを起動したり、トランザクション・ログが見つからないときにデータベース・サーバを起動したりする場合に使用します。-f はデータベース・オプションであり、サーバ・オプションではないことに注意してください。

  リカバリが完了したら、サーバを停止し、-f オプションを指定せずに再起動してください。

  「-f リカバリ・オプション」 217 ページを参照してください。

SQL Anywhere でのスレッド

SQL Anywhere のスレッド・モデルを理解するには、スレッドと要求処理の基本用語と概念も理解することが必要です。

- **要求**　**要求**は、クエリや SQL 文などの作業単位で、接続を介してサーバに送信されます。要求の存続期間は、要求がデータベースによって最初に受信されてから、結果の最後が返されてカーソルが閉じられるまで、または要求がキャンセルされるまでの間を指します。

- **タスク**　**タスク** は、データベース・サーバ内で実行されるアクティビティの単位です。また、サーバによってスケジュールされる最小の処理単位でもあります。ユーザ要求は、それぞれ、データベース・サーバ内で少なくとも 1 つのタスクになります。クエリ内並列処理が関連する場合は、複数のタスクになることもあります。データベース・サーバは、ユーザ要求だけでなく、内部的な管理処理を行うために自身のタスクをスケジュールすることもできます。たとえば、クリーナの実行やタイマの処理などがこれに該当します。同時に実行できるアクティブ・タスクの最大数は、-gn オプションで設定します。データベース・サーバで同時に処理できる数よりもタスクの数が多くなると、それらのタスクは実行待ちになります。アクティブ・タスクまたは処理がすでに開始しているタスクにおいて、ロックを待機したり、I/O 処理の完了を待機したりするなどの理由でブロックが必要になった場合でも、そのタスクはアクティブであると見なされます。このような理由で、上限数は -gn オプションの値で設定されます。

- **スレッド**　**スレッド** はオペレーティング・システムを構築するもので、アプリケーション内の**制御用のスレッド (thread-of-control)** を実行することを示します。データベース・サーバを含め、オペレーティング・システムの各プロセスは、少なくとも 1 つのスレッドによって実行されます。複数のスレッドによって実行されることもあります。スレッドは、オペレーティング・システムによってアプリケーション外部でスケジュールされ、最終的にアプリケーション実行のすべてはスレッドによって行われます。SQL Anywhere データベース・サーバ内にあるタスクは、オペレーティング・システム・スレッドで実行します。SQL Anywhere は、起動時に決められた数のスレッドを作成します。この数は、-gtc オプション (Windows または Linux の場合) または -gn オプション (UNIX または NetWare の場合) で制御されます。

参照

- 「スレッド動作の制御」 25 ページ
- 「クエリ実行時の並列処理」 『SQL Anywhere サーバ - SQL の使用法』
- 「-gn サーバ・オプション」 172 ページ
- 「-gtc サーバ・オプション」 175 ページ
- 「sa_clean_database システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
- 「トランザクションのブロックとデッドロック」 『SQL Anywhere サーバ - SQL の使用法』

UNIX と NetWare でのタスク

UNIX と NetWare では、タスクはオペレーティング・システム・スレッドで直接実行されます。これらのプラットフォームでは、オペレーティング・システム・スレッドの数は -gn オプション・セットの値によって設定されます。データベース・サーバの起動時、オプションで設定された値に基づいてスレッドが作成されます。すべてのタスクは、このスレッド・セットから実行されます。スレッドが利用可能になると、そのスレッドは、処理を要求するタスクの中から、次に利用可能なタスクを選択します。タスクの処理を開始したスレッドは、そのタスクが完了するまで存続します。タスクにおいて、I/O 操作やロックを待機するなどの理由でブロックが必要になると、スレッドは自発的に CPU の制御を解放してオペレーティング・システムのスケジューラに戻し、CPU で他のスレッドを実行できるようにします。

スレッドは、自発的に CPU を解放する場合もありますが、オペレーティング・システムのスケジューラによって先取りされることもあります。プロセス内の各アプリケーション・スレッドには、実行する一連のタイム・スライス、実行時間の長さ (優先度によって決定されます)、その他のシステム要因などが指定されています。スレッドが現在のタイム・スライスの最後に達すると、そのスレッドはオペレーティング・システムによって先取りされ、後で再び実行されるようにスケジュールされます。オペレーティング・システムのスケジューラは、別のスレッドを選択し、タイム・スライスに基づいてそのスレッドを実行します。この先取りスケジュールは、タスクの処理に対して、目に見える形で影響を及ぼすことはありません。スレッドが再び実行されるようにスケジュールされると、タスクは中断したところから処理が開始されます。

アクティブ・タスクの処理が完了すると、スレッドは、処理が可能なタスクが他にあるかどうかを確認します。該当するタスクが存在する場合は、スレッドは次に利用可能なタスクを選択し、処理を続けます。それ以外の場合は、スレッドは CPU を解放し、新しいタスクがデータベース・サーバに到着するのを待機します。

参照

- 「-gn サーバ・オプション」 172 ページ

Windows と Linux でのタスク

Windows と Linux では、タスクは**ファイバ**と呼ばれる軽量スレッドで実行されます。ファイバを使用すると、オペレーティング・システムのスレッドのスケジューラに依存せず、協調性を持ってスケジュールするように、スレッドでタスクを実行できます。オペレーティング・システムのカーネルやスケジューラの介入がないため、ファイバ間の切り替えでは、スレッドを使用した切り替えに比較して負荷が大幅に低下します。スレッドを使用した切り替えが頻繁に発生するマル

チスレッド・アプリケーションでファイバを使用すると、パフォーマンスとスケーラビリティを格段に向上させることができます。

オペレーティング・システム・スレッドに依存しないため、ファイバは、他のアクティビティの完了を待機しているときは、制御を別のファイバに明示的に譲ります。たとえば、ファイバで実行中のタスクにおいて、I/O 操作の完了を待機するなどの理由でブロックが必要になると、ファイバは制御を解放して他のファイバに譲ります。元のファイバを実行しているスレッドは、カーネルを介した切り替えを行うことなく、即座に別のファイバを選択して実行を開始できます。ファイバがブロックしたが制御を譲らなかった場合、そのファイバを実行しているスレッドがブロックされ、他のファイバがそのスレッドで実行できなくなります。複数のスレッドがファイバを実行している場合、待機中のファイバを実行しているスレッドだけがブロックされます。他のスレッドは、ファイバを自由に実行できます。

ファイバをサポートするプラットフォームには、少なくとも、サーバの最大同時性設定 (-gn オプションで指定) で求められた数と同数のファイバが存在します。内部のサーバ・タスクに対してファイバが常にサービスを提供できるように、サーバは指定された数よりも多いファイバを作成することもあります。「-gn サーバ・オプション」 172 ページを参照してください。

スレッド動作の制御

スレッドの動作は、主に 5 つの要素によって制御されます。これらは、サーバ・オプションによって管理されます。すべてのプラットフォームで、これらのオプションのすべてがサポートされるわけではありません。

- **マルチプログラミング・レベル (-gn サーバ・オプション)**
  -gn オプションは、サーバのマルチプログラミング・レベルを制御します。この値によって、同時にアクティブにできるタスクの最大数が決定されます。データベース要求は、それぞれ、少なくとも 1 つのタスクを使用します。クエリ内並列処理が関連する場合は、複数のタスクを使用することもあります。また、サーバは、内部的な管理処理を行うために臨時にタスクをスケジュールすることもあります。サーバ内にあるタスクの数がマルチプログラミング・レベルを超えると、未処理のタスクは、現在実行中のタスク (アクティブ・タスク) が完了するのを待機します。デフォルトでは、ネットワーク・データベース・サーバ用とパーソナル・データベース・サーバ用に最大で 20 のタスクを同時に実行できます。「-gn サーバ・オプション」 172 ページと「データベース・サーバのマルチプログラミング・レベルの設定」 27 ページを参照してください。

- **内部実行スレッドあたりのスタック・サイズ (-gss サーバ・オプション)**
  -gss オプションを使用して、サーバの内部実行スレッドあたりのスタック・サイズを設定できます。-gss オプションによって、データベース・サーバのメモリ使用量を節約できます。これは、メモリが限られている環境で役立ちます。このオプションは、Windows オペレーティング・システムでは無効です。「-gss サーバ・オプション」 174 ページを参照してください。

- **プロセッサの数 (-gt サーバ・オプション)**
  複数のプロセッサがある場合は、-gt オプションを指定して、スレッドが使用できるプロセッサの数を制御できます。このオプションは、NetWare ではサポートされていません。「-gt サーバ・オプション」 174 ページを参照してください。

- **プロセッサの同時実行性 (-gtc サーバ・オプション)**

CPU 上で同時に実行できるスレッド数の上限を指定できます。デフォルトでは、データベース・サーバは、ライセンスされた各物理プロセッサにおいて、すべてのハイパースレッドとコアで実行されます。このオプションは、NetWare ではサポートされていません。「-gtc サーバ・オプション」 175 ページを参照してください。

スレッド処理のヒント

- -gn を大きくすると、スレッドのデッドロックが発生する可能性を低減できます。「-gn サーバ・オプション」 172 ページを参照してください。
- -gt を 1 に設定すると、同時性の問題を回避するのに役立つことがあります。「-gt サーバ・オプション」 174 ページを参照してください。
- Windows で使用する -gn の値を決定する場合は、パフォーマンス・モニタで [要求：アクティブ] や [要求：未スケジュール] の値を確認すると役立ちます。アクティブな要求の数が常に -gn よりも少ない場合は、-gn を減らすことができます。要求の合計数 (アクティブ + 未スケジュール) が頻繁に -gn を上回る場合は、-gn の値を増やすことができます。「パフォーマンス・モニタの統計値」『SQL Anywhere サーバ - SQL の使用法』と「-gn サーバ・オプション」 172 ページを参照してください。

プロセッサの使用とスレッド処理の例

次の例は、データベース・サーバが -gt と -gtc をもとにどのように CPU を選択するかを示します。この例では、システムが 4 プロセッサ構成で、各プロセッサにコアが 2 つあることを前提としています。物理プロセッサは文字で、コアは数字でそれぞれ区別します。したがって、このシステムにはプロセッサ・ユニットとして A0、A1、B0、B1、C0、C1、D0、D1 が存在することになります。

シナリオ	ネットワーク・データベース・サーバの設定
シングル CPU ライセンス、または -gt に 1 が指定されている	◆ -gt 1 ◆ -gtc 2 ◆ -gn 20 スレッドは A0 または A1 で実行できます。
CPU のライセンスに制限がなく、-gtc に 5 が指定されている	◆ -gt 4 ◆ -gtc 5 ◆ -gn 20 スレッドは A0、A1、B0、C0、D0 で実行できます。
データベース・サーバに 3 CPU ライセンスがあり、-gtc 5 が指定されている	◆ -gt 3 ◆ -gtc 5 ◆ -gn 20 スレッドは A0、A1、B0、B1、C0 で実行できます。

シナリオ	ネットワーク・データベース・サーバの設定
CPU のライセンスに制限がなく、-gtc に 1 が指定されている	◆ -gt 4 ◆ -gtc 1 ◆ -gn 20 スレッドは A0 でのみ実行できます。

データベース・サーバのマルチプログラミング・レベルの設定

データベース・サーバの「**マルチプログラミング・レベル**」は、同時にアクティブにできるタスクの最大数です。-gn サーバ・オプションの設定によって制御されます。アクティブ・タスクは、データベース・サーバにおいて 1 つのスレッド (またはファイバ) によって現在実行されているタスクを指します。アクティブ・タスクは、アクセス・プランの演算子を実行したり、その他の実質的な処理を行ったりします。ただし、(I/O 操作やロー・ロックなどのために) リソースを待機することもあります。未スケジュールのタスクは、実行の準備が整っているものの、利用可能なスレッドやファイバを待機しているタスクを指します。同時に実行できるアクティブ・タスクの数は、データベース・サーバのスレッドの数と、コンピュータに搭載されている論理プロセッサの数によって異なります。

マルチプログラミング・レベルは、サーバの実行中は一定に維持され、サーバ上のすべてのデータベースに適用されます。アクティブ・タスクのデフォルト数は、ネットワーク・データベース・サーバおよびパーソナル・データベース・サーバでは 20 ですが、Windows CE のデフォルトは 3 です。

マルチプログラミング・レベルの増加

マルチプログラミング・レベルをどのタイミングで高めたり低くしたりするかを決定するのは、難しいことがあります。たとえば、データベース・アプリケーションが Java ストアド・プロシージャを利用する場合や、クエリ内並列処理が有効になっている場合、これらの要求を処理するために追加で作成されたサーバ・タスクは、マルチプログラミングの制限を超えると、他の要求が完了するまで実行を待機することになります。このような状況では、マルチプログラミング・レベルを高めることが適切です。多くの場合、マルチプログラミング・レベルを高めると、データベース・サーバの全体的なスループットも相応して向上します。これは、マルチプログラミング・レベルを高めることによって同時に実行できるタスク (要求) の数が増えるためです。ただし、マルチプログラミング・レベルを高めるときに検討が必要なトレードオフがあります。これには次のようなものがあります。

- **競合の増加**　同時に実行するタスクの数を増加すると、アクティブな要求間で競合が発生する可能性が高まることがあります。競合はリソースに関連するもので、スキーマ・ロックやロー・ロック、データベース・サーバ内部のデータ構造や同期プリミティブなどが挙げられます。このような状況では、サーバのスループットが実際に低下することがあります。
- **サーバのオーバヘッドの増加**　各アクティブ・タスクは、スレッドの割り付けと管理を要求します (Windows と Linux の場合は、軽量スレッドがファイバを呼び出します)。スケジュールを制御するために追加の管理構造も必要です。さらに、各アクティブ・タスクは、実行スタックに対するアドレス領域の事前割り付けを要求します。スタック・サイズはプラットフォームによって異なりますが、32 ビットのプラットフォームで約 1 MB、64 ビットのプラットフォー

ムではより大きなサイズが必要です。Windows システムでは、スタック領域の割り付けはサーバ・プロセスのアドレス領域に影響を及ぼします。ただし、スタック・メモリは要求に応じて割り付けられます。UNIX プラットフォームの場合 (Linux を含みます)、スタックのバッキング・メモリが即座に割り付けられます。したがって、マルチプログラミング・レベルを高めると、サーバのメモリ・フットプリントが増加します。そして、利用できるアドレス領域が少なくなるため、キャッシュで利用できるメモリ量が減少します。

- **スラッシング**　データベース・サーバは、個々の要求に対する処理を実行するためではなく、実行のオーバヘッドを管理するためだけに、多大な量のリソースを使用する状態に陥ることがあります。この状態は一般的に「スラッシング」と呼ばれています。たとえば、データベース・キャッシュ内の領域に対して、対立するアクティブな要求の数が多すぎるのに、これらの要求のセットが使用するデータベース・ページのワーキング・セットを収容するだけの十分な大きさをキャッシュが備えない場合などに、このような状態になります。結果として、オペレーティング・システムで発生する場合と同様に、ページの横取りが発生することがあります。
- **クエリ処理への影響**　データベース・サーバは、-gn オプションで指定されたタスクの最大数を処理するために、常に十分なリソースが存在する状態を確保する必要があります。最も重要なリソースはメモリです。ハッシュ・ジョインやソートなどのクエリ実行演算子は、効率的に実行するために大量のメモリを要求することがあります。

各タスクには、そのタスクに使用されるメモリ量を制限するガバナーが存在します。ソフトによる制限は、次の式によって定義されます。

(total size of the database cache) / (multiprogramming level)

しきい値を超えるタスクが存在する場合、データベース・カーネルは、該当するタスクに対してクエリ実行演算子を要求し、(可能であれば) 不要なメモリ・バッファを解放するように要求します。これは、制限値を超えなくなるようにするための措置です。結果として、実行演算子は、実行時のメモリ量が少なくなるような方式に切り替えられます。これによって、実行時間は長くなりますが、使用するメモリ量を節約できます。

クエリ・オプティマイザは、アクセス・プランのコスト計算時にメモリのしきい値を考慮し、しきい値を超える量のメモリに依存する実行方式は選択しないようにします。また、各タスクには次のような制限が設定されます。

(1/4 maximum cache size) / (number of currently active tasks)

制限を超えると、文にはエラーが発生します。

マルチプログラミング・レベルの低下

同時に実行するタスクの数を減らしてデータベース・サーバのマルチプログラミング・レベルを低く設定すると、一般に、サーバのスループットは低下します。ただし、マルチプログラミング・レベルを低くすることで、個々の要求の応答時間は短縮されることがあります。これは、リソースに対して競合する要求の数が少なくなり、ロック競合の確率が低下するためです。

SQL Anywhere では、スレッド (ファイバ) は協調性のある方法でタスクを実行します。あるタスクが完了すると、スレッド (ファイバ) は、実行を待機している次のタスクを選択できます。ただし、ロー・ロックを待機している場合など、タスクがブロックされていると、スレッド (ファイバ) もブロックされます。

設定したマルチプログラミング・レベルが低すぎると、**スレッド・デッドロック**が発生することがあります。データベース・サーバに *n* スレッド (ファイバ) を設定した場合を考えてみます。*n*-1 の数のスレッドがブロックされているときに最後のスレッドをブロックしようとすると、スレッド・デッドロックが発生します。データベース・サーバのカーネルは、最後のスレッドをブロックすることを許可できません。ブロックすることによって、すべてのスレッドがブロックされ、サーバがハングするためです。ブロックを許可する代わりに、データベース・サーバは、最後のスレッドをブロックしようとしたタスクを終了します。このとき、SQLSTATE 値には 40W06 が設定されます。

マルチプログラミング・レベルが負荷に対して適切なレベルである場合に、スレッド・デッドロックが発生するときは、アプリケーション設計に問題があることが考えられます。結果として競合が発生し、延いてはスケーラビリティが低下します。例として、データベースに新しいデータを挿入するときにすべてのアプリケーションが変更を加える必要のあるテーブルについて考えてみます。この方法は、プライマリ・キーを生成するスキームの一部として、よく使用されます。このようなテーブルでは、結果として、アプリケーションの挿入トランザクションのすべてが事実上直列化される状況になります。共有テーブルが直列化されることに起因して、サーバがサービスを提供する速度よりも挿入トランザクション速度の方が高くなると、通常、スレッド・デッドロックが発生します。

マルチプログラミング・レベルの選択

アプリケーションの負荷を測定し、サーバ・スループットと要求の応答時間に対する、サーバのマルチプログラミング・レベルの効果を分析することをおすすめします。プロパティ機能と同様に、さまざまなパフォーマンス カウンタを利用できます。Windows 環境でアプリケーション・テスト時にデータベース・サーバ動作を分析するには、Windows パフォーマンス モニタが役立ちます。この分析には、アクティブな要求や未スケジュールの要求に関連のあるパフォーマンス カウンタが特に重要です。

アクティブな要求の数が、-gn データベース・サーバ・オプションの値よりも常に小さくなる場合は、マルチプログラミング・レベルを低く設定することを検討できます。ただし、クエリ内並列処理 (サーバの実行キューに対してタスクを追加します) の効果を考慮に入れる必要があります。クエリ内並列処理の効果が限界的である場合は、全体的なシステム・スループットを低下させることなく、安全に、マルチプログラミング・レベルを低く設定できます。要求の総数 (アクティブな要求 + 未スケジュールの要求) が -gn データベース・サーバ・オプションの値よりも大きくなることが多い場合は、上に概要を示したトレードオフを前提として、マルチプログラミング・レベルを高く設定できます。パフォーマンス・モニタは、NetWare、UNIX、または Linux プラットフォームでは使用できないことに注意してください。

通信プロトコルの選択

クライアント・アプリケーションとデータベース・サーバ間の通信では、通信プロトコルが必要です。SQL Anywhere は、ネットワーク通信用と同一コンピュータ通信用の通信プロトコル・セットをサポートします。

デフォルトでは、データベース・サーバは、使用可能なプロトコルをすべて起動します。-x オプションを使用すると、データベース・サーバで使用できるプロトコルを制限できます。クライアント側では、CommLinks (LINKS) 接続パラメータを使用してほとんど同じオプションを制御できます。

これらのオプションを使用したサーバの実行については、「サポートされているネットワーク・プロトコル」 122 ページを参照してください。

パーソナル・サーバで使用できるプロトコル

パーソナル・データベース・サーバ (*dbeng10.exe*) は、次のプロトコルをサポートします。

- **共有メモリ**　このプロトコルは、同一コンピュータ通信で使用され、常に使用可能です。すべてのプラットフォームで使用できます。
- **TCP/IP**　このプロトコルは、TDS クライアント、Open Client、または jConnect JDBC ドライバからの同一コンピュータ通信のみで使用されます。Open Client または jConnect から接続する場合は、TCP/IP を実行してください。

  TDS クライアントの詳細については、「Open Server としての SQL Anywhere」 985 ページを参照してください。

ネットワーク・サーバで使用できるプロトコル

ネットワーク・データベース・サーバ (*dbsrv10.exe*) は、次のプロトコルをサポートします。

- **共有メモリ**　このプロトコルは、同一コンピュータ通信で使用され、常に使用可能です。すべてのプラットフォームで使用できます。
- **SPX**　このプロトコルは、UNIX を除くすべてのプラットフォームでサポートされます。
- **TCP/IP**　このプロトコルは、すべてのプラットフォームでサポートされます。

共有メモリとターミナル・サービス
ターミナル・サービスを使用している場合、共有メモリ・クライアントが検索できるのは同じターミナルを実行しているデータベース・サーバのみです。サービスとして実行しているデータベース・サーバと組み合わせてターミナル・サービスを使用している場合は、コンソールで実行されているクライアントだけが接続可能であり、コンソールではないターミナルで実行中のクライアントは共有メモリを使用して接続することはできません。この場合、共有メモリを TCP/IP の代わりに使用すると、クライアントが接続できるようになります。

UNIX の共有メモリ接続の保護の詳細については、「セキュリティのヒント」 920 ページを参照してください。

プロトコルの指定

-x オプションを使用すると、使用可能なネットワーク・プロトコルの一部だけを使用するようにデータベース・サーバに指示できます。次のコマンドは、TCP/IP プロトコルを使用するサンプル・データベースを起動します。

```
dbsrv10 -x "tcpip" samples-dir¥demo.db
```

引用符はこの例では必須ではありませんが、-x の引数にスペースがある場合は必要です。

補足パラメータを追加して、プロトコルごとにサーバの動作をチューニングできます。たとえば、次のコマンド (すべてを 1 行に入力) は、2 つのネットワーク・カード (そのうち 1 つは指定したポート番号を持つ) を使用するように指示します。

```
dbsrv10 -x "tcpip(MyIP=192.75.209.12:2367,192.75.209.32)" samples-dir¥demo.db
```

-x オプションとともに使用できるネットワーク・プロトコル・オプションについては、「ネットワーク・プロトコル・オプション」 265 ページを参照してください。

データベース・サーバの停止

データベース・サーバは、次のいずれかの方法で停止します。

- サーバ・メッセージ・ウィンドウで [シャットダウン] をクリックする。
- dbstop ユーティリティを使用する。

  dbstop ユーティリティは、バッチ・ファイルの中で使用するか、別のコンピュータのサーバを停止するときに特に便利です。これはコマンドに接続文字列が必要です。「サーバ停止ユーティリティ (dbstop)」 732 ページを参照してください。
- アプリケーションの切断時にデフォルトで自動停止するようにする(この方法は、アプリケーション接続文字列で起動したパーソナル・サーバの場合のみ使用できます)。
- UNIX コンピュータまたは NetWare コンピュータのサーバ・メッセージ・ウィンドウが前面に表示されているときに [Q] キーを押す。

例

- **dbstop ユーティリティを使用してサーバを停止するには、次の手順に従います。**

1. サーバを起動します。たとえば、SQL Anywhere インストール・ディレクトリから実行される次のコマンドは、サンプル・データベースを使用する Ottawa という名前のサーバを起動します。

   ```
   dbsrv10 -n Ottawa samples-dir¥demo.db
   ```

2. dbstop を使用してサーバを停止します。

   ```
   dbstop -c "ENG=Ottawa;UID=DBA;PWD=sql"
   ```

サーバを停止できるユーザ

サーバの起動時に -gk オプションを使用すると、ユーザが dbstop を使用してサーバを停止するために必要なパーミッション・レベルを設定できます。パーソナル・データベース・サーバの場合、デフォルト設定は all です。ネットワーク・データベース・サーバの場合、必要なパーミッションのデフォルト・レベルは DBA ですが、この値を all または none に設定することもできます (もちろん、そのコンピュータを使用していれば誰でもサーバ・メッセージ・ウィンドウで [シャットダウン] をクリックできます)。

オペレーティング・システム・セッションの停止

データベース・サーバが実行中のオペレーティング・システム・セッションを閉じたり、オペレーティング・システム・コマンドを使用してデータベース・サーバを停止すると、サーバは正しく停止しません。そのような操作をすると、次回データベースをロードしたときにリカバリが必要になります。このリカバリは自動的に行われます。

リカバリの詳細については、「バックアップとデータ・リカバリ」 811 ページを参照してください。

データベース・サーバを明示的に停止してから、オペレーティング・システムのセッションを閉じることをおすすめします。ただし NetWare では、NetWare サーバ・コンピュータを正しく停止すると、データベース・サーバが正しく停止します。

次に、サーバを正しく停止しないコマンド例を示します。

- Windows の [タスク マネージャ] でプロセスを停止する
- UNIX の slay または kill コマンドを使用する

データベースの開始と停止

データベース・サーバは、一度に複数のデータベースをロードできます。次のコマンドで、データベースとサーバを同時に起動できます。

```
dbeng10 demo sample
```

警告
データベース・ファイルは、データベース・サーバと同じコンピュータ上に置いてください。ネットワーク・ドライブにあるデータベース・ファイルを操作すると、ファイルが破損することがあります。

起動中のサーバ上でのデータベースの開始

次のいずれかの方法で、サーバの起動後にもデータベースを起動できます。

- サーバに接続されているときに、DatabaseFile (DBF) 接続パラメータを使用してデータベースに接続する。DatabaseFile (DBF) 接続パラメータは、新規接続用のデータベース・ファイルを指定します。そのデータベース・ファイルが現在のサーバ上で起動します。

  「組み込みデータベースへの接続」 68 ページまたは「DatabaseFile 接続パラメータ [DBF]」 240 ページを参照してください。

- サーバが選択されているときに、START DATABASE 文を使用するか、Sybase Central で [ファイル] – [データベースの開始] を選択します。

  「START DATABASE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

制限事項

- サーバは、固定サイズのページを使用して、データベース情報をメモリに保持します。サーバがいったん起動すると、サーバ・ページよりも大きいページ・サイズのデータベースは起動できません。

  「最大ページ・サイズの設定」 22 ページを参照してください。

- -gd サーバ・オプションは、データベースの開始に必要なパーミッションを決定します。

データベースの停止

データベースは、次のいずれかの方法で停止します。

- 接続文字列で、起動したデータベースとの接続を切断する。AutoStop (ASTOP) 接続パラメータを明示的に NO に設定しないかぎり、この動作が自動的に行われます。

  「AutoStop 接続パラメータ [ASTOP]」 233 ページを参照してください。

- Interactive SQL または Embedded SQL で STOP DATABASE 文を使用します。

「STOP DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

現在のセッション外でのサーバの起動

ユーザ ID とパスワードを使用してコンピュータにログインすると、ユーザは「**セッション**」を確立します。データベース・サーバ、またはその他のアプリケーションを起動すると、そのセッション内で稼働します。コンピュータからログオフすると、そのセッションに関連するすべてのアプリケーションは終了します。

一般的に、データベース・サーバは常に使用可能である必要があります。これを簡単に実現するには、Windows または UNIX 用の SQL Anywhere を、ユーザがコンピュータからログオフしてもデータベース・サーバが稼働し続けるようにします。これを行う方法は、使用するオペレーティング・システムによって異なります。

- **Windows サービス**　Windows データベース・サーバをサービスとして実行できます。サービスには、高可用性サーバを実行するための便利な特性が多数あります。「Windows サービスの概要」 38 ページを参照してください。
- **UNIX デーモン**　-ud オプションを使用すると、UNIX データベース・サーバをデーモンとして実行できます。これによって、データベース・サーバをバックグラウンドで実行し、ログオフ後も引き続き実行させることができます。「デーモンとしての UNIX データベース・サーバの稼働」 36 ページを参照してください。
- **Linux サービス**　Linux データベース・サーバをサービスとして実行できます。サービスには、高可用性サーバを実行するための便利な特性が多数あります。「Linux 用サービス・ユーティリティ (dbsvc)」 711 ページを参照してください。

SQL Anywhere データベース・サーバ用のサービスを作成するだけでなく、次の実行プログラム用に Windows サービスを作成することもできます。

- SQL Anywhere Log Transfer Manager (LTM)
- SQL Remote Message Agent (dbremote)
- Mobile Link サーバ (mlsrv10)
- Mobile Link 同期クライアント (dbmlsync)
- SQL Anywhere Broadcast Repeater (dbns10)
- Listener ユーティリティ (dblsn)

参照

- 「Windows 用サービス・ユーティリティ (dbsvc)」 704 ページ

デーモンとしての UNIX データベース・サーバの稼働

UNIX データベース・サーバをバックグラウンドで実行し、現在のセッションから独立して稼働させるには、データベース・サーバを「**デーモン**」として起動します。

データベース・サーバをバックグラウンドで実行するのに '&' を使用しない
データベース・サーバをバックグラウンドで実行するのに UNIX の & (アンパサンド) コマンドを使用しても機能しません。サーバが応答しなくなります。データベース・サーバはデーモンとして実行してください。
同様に、典型的な fork()-exec() シーケンスを使用してプログラムの中からサーバをバックグラウンドで起動しようとしても動作しません。

UNIX データベース・サーバは、次のいずれかの方法でデーモンとして実行できます。

1. データベース・サーバの起動時に、-ud オプションを使用する。次に例を示します。

   ```
   dbsrv10 -ud demo
   ```

2. dbspawn ツールを使用してデータベース・サーバを起動する。次に例を示します。

   ```
   dbspawn dbsrv10 demo
   ```

   dbspawn を使用することの利点は、デーモンが起動し、要求を受け入れる状態になったことを確認するまで dbspawn プロセスが終了しないことです。何らかの理由でデーモンの起動が失敗した場合、dbspawn の終了コードは 0 以外の値になります。

   -ud オプションを使用してデーモンを直接起動したときは、dbeng10 コマンドと dbsrv10 コマンドがデーモン・プロセスを作成し、(終了して次のコマンドを実行できるように) すぐに返します。その後、デーモンがそれ自体を初期化するか、コマンドで指定されたデータベースを開こうとします。

   データベース・サーバを使用するアプリケーションでデーモンの実行を確実にしたい場合、アプリケーションの起動前にデーモンを実行する dbspawn を使用します。次の例では、csh スクリプトを使用してこれをテストする方法を示します。

   ```
   #!/bin/csh
   # start the server as a daemon and ensure that it is
   # running before you start any applications
   dbspawn dbsrv10 demo
   if ( $status != 0 ) then
     echo Failed to start demo server
     exit
   endif
   # ok, now you can start the applications
   ...
   ```

   次の例では、sh スクリプトを使用して、アプリケーションの起動前にデーモンが実行中であるかどうかをテストします。

   ```
   #!/bin/sh
   # start the server as a daemon and ensure that it is
   # running before you start any applications
   dbspawn dbsrv10 demo
   if [ $? != 0 ]; then
     echo Failed to start demo server
     exit
   fi
   # ok, now you can start the applications
   ...
   ```


3. C プログラムの中からデーモンを生成する。次に例を示します。

```
...
if( fork() == 0 ) {
     /* child process = start server daemon */
     execl( "/opt/sqlanywhere10/bin/dbsrv10",
"dbsrv10", "-ud", "demo" );
  exit(1);
}
/* parent process */
...
```

-ud オプションが使用されていることに注意してください。

参照

- 「-ud サーバ・オプション」 201 ページ
- 「サーバ・バックグラウンド起動ユーティリティ (dbspawn)」 730 ページ

Windows サービスの概要

データベース・サーバは、サービスではなく、他の Windows プログラムと同じように稼働できますが、マルチユーザ環境で標準プログラムとして実行する場合は特に、制限があります。

標準実行プログラムとして実行する場合の制限事項

プログラムを起動すると、プログラムは Windows のログイン・セッションで動作します。これは、コンピュータをログオフするとプログラムが停止することを意味します。データベース・サーバで通常行われるように、プログラムを常に動作させておきたい場合、このことはコンピュータの用途を制限します。データベース・サーバを実行し続けるには、そのデータベース・サーバを実行するコンピュータにログオンし続けます。これは、Windows コンピュータをログオンした状態にしておくことになるので、セキュリティ上の問題も発生します。

サービスの利点

アプリケーションを Windows サービスとしてインストールすると、ログオフしても実行できます。

サービスを開始するときに、特別なシステム・アカウントの LocalSystem (または指定されている別のアカウント) を使ってログオンします。サービスを起動するユーザの ID とサービスは関連していないので、起動した人がログオフしてもサービスは開いたままになります。Windows コンピュータが起動してユーザがログオンする前に、サービスを自動的に開始するように設定することもできます。

サービスの管理

Sybase Central は、Windows のサービス マネージャよりも便利でわかりやすい方法で SQL Anywhere サービスを管理します。dbsvc ユーティリティを使用してサービスの作成や変更を行うこともできます。「Windows 用サービス・ユーティリティ (dbsvc)」 704 ページを参照してください。

Windows サービスとして実行できるプログラム

サービスとして実行できるプログラムは次のとおりです。

- ネットワーク・データベース・サーバ (*dbsrv10.exe*)
- パーソナル・データベース・サーバ (*dbeng10.exe*)
- SQL Anywhere Broadcast Repeater ユーティリティ (*dbns10.exe*)
- Mobile Link サーバ (*mlsrv10.exe*)
- SQL Remote Message Agent (*dbremote.exe*)
- Log Transfer Manager ユーティリティ (*dbltm.exe*)
- Listener ユーティリティ (*dblsn.exe*)
- Mobile Link 同期クライアント (*dbmlsync.exe*)

これらのアプリケーションすべてが、SQL Anywhere のすべてのエディションに付属しているわけではないことに注意してください。

Windows サービスの管理

コマンド・ラインから、または Sybase Central の [サービス] タブで、以下の Windows サービス管理タスクを実行できます。

- サービスの作成、編集、削除
- サービスの開始と停止
- サービスを制御するパラメータの変更
- サービスにデータベースを追加して、同時に複数のデータベースを実行できるようにする

Sybase Central のサービス・アイコンは、サービスが実行中であるか停止されているかを示すアイコンを使用して、各サービスの現在の状況を表示します。

Windows サービスの作成

この項では、Sybase Central とサービス・ユーティリティを使用してサービスを設定する方法について説明します。

- **新しいサービスを作成するには、次の手順に従います (Sybase Central の場合)。**

1. 左ウィンドウ枠で、SQL Anywhere 10 プラグインを選択します。
2. 右ウィンドウ枠にある [サービス] タブをクリックします。
3. [ファイル] メニューから [新規] – [サービス] を選択します。

   [サービス作成] ウィザードが表示されます。
4. ウィザードの指示に従います。

ヒント
Mobile Link プラグインのサービスを作成することもできます。「現在のセッション外での Mobile Link サーバの起動」 『Mobile Link - サーバ管理』を参照してください。

- **新しいサービスを作成するには、次の手順に従います (コマンド・ラインの場合)。**

1. コマンド・プロンプトを開きます。
2. -w オプションを使用して、サービス・ユーティリティを実行します。

   たとえば、myserv というパーソナル・サーバ・サービスを作成します。データベース・サーバを LocalSystem ユーザとして実行します。次のコマンドを入力します。

   ```
   dbsvc -as -w myserv "c:¥Program Files¥SQL Anywhere 10¥win32¥dbeng10.exe" -n william -c 8m "c:¥temp¥sample.db"
   ```

「Windows 用サービス・ユーティリティ (dbsvc)」 704 ページを参照してください。

注意

- サービス名の最初の 8 文字は、ユニークにしてください。
- サービスを自動で開始するよう選択すると、コンピュータが Windows を起動するときに必ずサービスが開始されます。手動で開始することを選択した場合は、毎回 Sybase Central から開始する必要があります。今後使用するためにサービスを設定している場合は、[無効] を選択します。
- Sybase Central でサービスを作成する場合は、実行プログラム自体の名前ではなく、実行プログラムのオプションをウィンドウで入力します。たとえば、20 MB のキャッシュ・サイズで myserver という名前を使用したサンプル・データベースを指定して、ネットワーク・サーバを実行する場合は、Sybase Central の [サービス作成] ウィザードの [パラメータ] ボックスに次のように入力します。

  ```
  -c 20M
  -n myserver samples-dir¥demo.db
  ```

  改行は任意です。
- サービスを実行するアカウントを選択します (LocalSystem という特別なアカウントまたは別のユーザ ID)。

  この選択の詳細については、「アカウント・オプションの設定」 43 ページを参照してください。
- Windows のデスクトップからサービスにアクセスできるようにする場合は、[デスクトップとの対話をサービスに許可] チェックボックスをオンにします。このチェックボックスをオフにすると、アイコンはシステム・トレイに表示されず、ウィンドウもデスクトップに表示されません。

「Windows サービスの設定」 41 ページを参照してください。

Windows サービスの削除

サービスを削除すると、サービスのリストからサーバ名が削除されます。サービスを削除しても、ハード・ディスクからソフトウェアが削除されることはありません。

前に削除したサービスを再インストールするには、オプションを再入力する必要があります。

♦ **Windows サービスを削除するには、次の手順に従います (Sybase Central の場合)。**

1. コンテキスト・ドロップダウン・リストから [SQL Anywhere 10] プラグインを選択します。

   右ウィンドウ枠で、[サービス] タブをクリックします。
2. 右ウィンドウ枠で、削除するサービスを選択し、[編集] メニューで [削除] を選択します。

♦ **Windows サービスを削除するには、次の手順に従います (コマンド・ラインの場合)。**

1. コマンド・プロンプトを開きます。
2. -d オプションを使用して、サービス作成ユーティリティを実行します。

   たとえば、myserv というサービスを確認プロンプトを表示させずに削除するには、次のコマンドを入力します。

   ```
   dbsvc -y -d myserv
   ```

参照

♦ 「Windows 用サービス・ユーティリティ (dbsvc)」 704 ページ

Windows サービスの設定

サービスは、一連のオプションを使ってデータベース・サーバやその他のアプリケーションを実行します。

サービスは、オプションに加えて、サービスが実行されるアカウントを指定するパラメータと、起動時の条件を指定するパラメータを許可します。

♦ **サービスのパラメータを変更するには、次の手順に従います。**

1. コンテキスト・ドロップダウン・リストから [SQL Anywhere 10] プラグインを選択します。
2. 右ウィンドウ枠で、変更するサービスを選択します。
3. [ファイル] – [プロパティ] を選択します。
4. [サービス] プロパティ・シートのタブで、必要に応じてパラメータを変更します。
5. 変更が完了したら [OK] をクリックします。

サービス設定の変更は、次回のサービス実行時から有効となります。起動オプションは、次回 Windows を起動するときに適用されます。

起動オプションの設定

次のオプションは、SQL Anywhere サービスの起動時の動作を制御します。これらのオプションは、サービスのプロパティ・シートの [一般] タブで設定できます。

- **[自動]** [自動] 設定を選択すると、サービスは Windows オペレーティング・システムが起動すると必ず起動します。この設定は、データベース・サーバやその他の常に稼働しているアプリケーションに適しています。
- **[手動]** [手動] 設定を選択すると、サービスは Administrator 権限を持つユーザが起動したときにのみ起動します。Administrator 権限については、Windows のマニュアルを参照してください。
- **[無効]** [無効] 設定を選択すると、サービスは起動しません。

オプションの指定

[サービス] プロパティ・シートの [設定] タブに、サービスのオプションを指定するための [パラメータ] テキスト・ボックスが用意されています。*このボックスには、実行プログラムの名前は入力しないでください。*

例

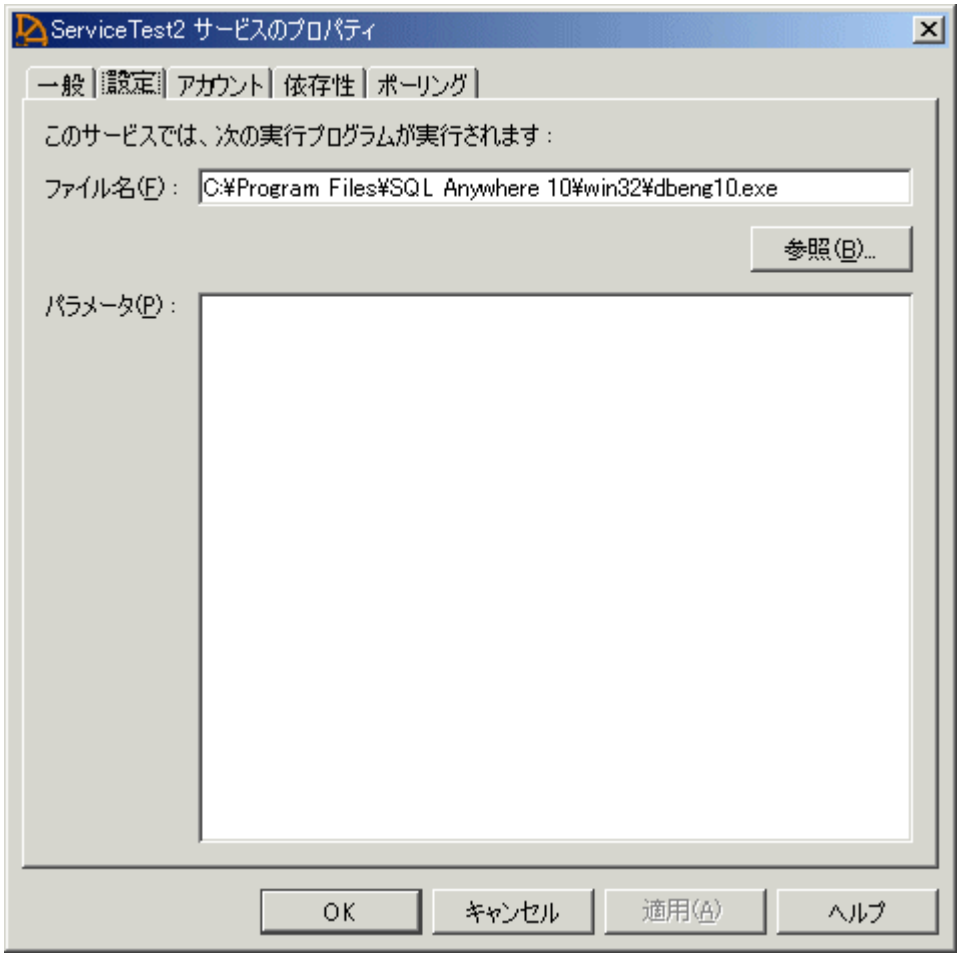
- 2 つのデータベースを実行する my_server というネットワーク・サーバ・サービスを、キャッシュ・サイズ 20 MB で起動するには、[パラメータ] フィールドに次のように入力します。

```
-c 20M
-n my_server
c:¥db_1.db
c:¥db_2.db
```

- サンプル・データベースにユーザ ID DBA で接続する SQL Remote Message Agent サービスを起動するには、次のように入力します。

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

次の図は、サービス・プロパティ・シートのサンプルを示します。



サービスのオプションは、実行プログラムのものと同じです。

アカウント・オプションの設定

サービスが実行されるアカウントを選択できます。ほとんどのサービスは、特別なアカウントの LocalSystem で実行され、これがサービスのデフォルト・オプションになっています。[サービス] プロパティ・シートの [アカウント] タブを開き、アカウント情報を入力して、別のアカウントでログオンするようにサービスを設定できます。

LocalSystem 以外のアカウントでサービスを実行するには、そのアカウントにサービスとしてログオンする権限が必要です。この権限は、Windows のユーザー マネージャの [高度なユーザー権利] で付与できます。この権限は、必要に応じて、サービス・ユーティリティ (dbsvc) を使って付与することもできます。

システム・トレイにアイコンが表示される場合

- サービスが LocalSystem で実行されているときに、[サービス] プロパティ・シートの [デスクトップとの対話をサービスに許可] がオンになっている場合は、サービスを実行するコンピュータ上の Windows にどのユーザがログインしてもデスクトップにアイコンが表示されます。したがって、すべてのユーザがアプリケーション・ウィンドウを開き、サービスとして実行されているプログラムを停止できます。
- サービスが LocalSystem で実行されているときに、[サービス] プロパティ・シートの [デスクトップとの対話をサービスに許可] がオフになっている場合は、ユーザのデスクトップにアイコンは表示されません。サービスの状態を変更する権限を与えられているユーザのみ、サービスを停止できます。
- サービスが他のアカウントで実行されている場合、デスクトップにアイコンは表示されません。サービスの状態を変更する権限を与えられているユーザのみ、サービスを停止できます。

実行ファイルの変更

サービスに関連するプログラム実行ファイルを変更するには、[サービス] プロパティ・シートの [設定] タブをクリックし、[ファイル名] ボックスに新しいパスとファイル名を入力します。

実行ファイルを新しいディレクトリに移動する場合は、この内容も変更します。

新しいデータベースをサービスに追加する

各ネットワーク・サーバまたはパーソナル・サーバは、複数のデータベースを実行できます。複数のデータベースを同時に実行するときは、新しいサービスを作成するのではなく、既存のサービスに新しいデータベースを付加することをおすすめします。

♦ 新しいデータベースを既存のサービスに追加するには、次の手順に従います。

1. コンテキスト・ドロップダウン・リストから [SQL Anywhere 10] プラグインを選択します。
2. 右ウィンドウ枠で、[サービス] タブをクリックします。
3. サービスを選択し、[ファイル] メニューから [プロパティ] を選択します。
4. [設定] タブをクリックします。
5. 新しいデータベースのパスとファイル名を、[パラメータ] ボックスのオプション・リストの最後に追加します。
6. [OK] をクリックして変更を保存します。

   次回サービスを開始したときに、新しいデータベースが起動します。

稼働中のサーバ上でデータベースを起動するには、Interactive SQL などのクライアント・アプリケーションを使います。

Interactive SQL からデータベースを起動する場合の詳細については、「START DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

Embedded SQL アプリケーションでこの機能を実装する方法については、「db_start_database 関数」『SQL Anywhere サーバ - プログラミング』を参照してください。

アプリケーションからデータベースを起動しても、サービスにはアタッチされません。サービスを停止して再起動すると、追加のデータベースは自動的に起動しません。

サービスのポーリング頻度の設定

Sybase Central では、指定した間隔でポーリングを行って、各サービスの状況 (開始または停止) をチェックし、アイコンを更新して現在の状況を表示できます。デフォルトでは、ポーリングはオフになっています。ポーリングをオフのままにしておくと、ステータスの変更を確認するには [フォルダの再表示] をクリックしなくてはなりません。

♦ **Sybase Central のポーリング頻度を設定するには、次の手順に従います。**

1. コンテキスト・ドロップダウン・リストから [SQL Anywhere 10] プラグインを選択します。
2. 右ウィンドウ枠で、[サービス] タブをクリックします。
3. サービスを選択し、[ファイル] メニューから [プロパティ] を選択します。
4. [ポーリング] タブをクリックします。
5. [ポーリングを可能にする] を選択します。
6. ポーリング頻度を設定します。

   頻度は選択したサービスだけでなく、すべてのサービスに適用されます。このウィンドウで設定した値は、変更するまでそのまま使われます。
7. [OK] をクリックします。

サービスの開始と停止

♦ **サービスを開始または停止するには、次の手順に従います。**

1. コンテキスト・ドロップダウン・リストから [SQL Anywhere 10] プラグインを選択します。
2. 右ウィンドウ枠で、[サービス] タブをクリックします。
3. サービスを選択し、[ファイル] メニューから [起動] または [停止] を選択します。

サービスを開始すると、停止するまで実行を続けます。Sybase Central を閉じたり、ログオフしてもサービスは停止しません。

データベース・サーバのサービスを停止すると、データベースへの接続はすべて閉じられ、データベース・サーバは停止されます。他のアプリケーションのプログラムは終了します。

Windows サービス マネージャ

SQL Anywhere のすべてのサービス管理は、Sybase Central から行うことができます。Windows の [コントロール パネル] にあるサービス マネージャを使用していくつかのタスクを実行することはできますが、Windows のサービス マネージャで SQL Anywhere サービスをインストールしたり設定したりすることはできません。

Windows の [コントロール パネル] から [サービス マネージャ] を開くと、サービスのリストが表示されます。SQL Anywhere サービスの名前は、サービスをインストールしたときに入力したサービス名に、SQL Anywhere がプレフィクスを付けたものです。インストールされたすべてのサービスが、リストに表示されます。

一度に複数のサービスを実行する

この項では、一度に複数のサービスを実行する方法について説明します。

サービスの依存

状況によっては、複数の実行プログラムをサービスとして実行する必要があり、これらの実行プログラムが相互に依存する場合があります。たとえば、レプリケーションを補助するために、サーバと SQL Remote Message Agent または Log Transfer Manager を実行する場合があります。

このような場合には、サービスを正しい順序で開始する必要があります。サーバが起動する前に SQL Remote Message Agent サービスが開始されると、サーバを検出できないためエラーになります。

「**サービス・グループ**」を使用すると、これらの問題を防ぐことができます。サービス・グループは Sybase Central で管理します。

サービス・グループの概要

システムの各サービスを、サービス・グループの各メンバに割り当てることができます。デフォルトでは、次の表にリストしたように各サービスがグループに属しています。

サービス	デフォルトのグループ
ネットワーク・サーバ	SQLANYServer
パーソナル・サーバ	SQLANYEngine
Mobile Link 同期クライアント	SQLANYMLSync
Replication Agent	SQLANYLTM

サービスが適切なグループのメンバであることをチェックしてから、サービスが正しい順序で開始するように設定してください。Sybase Central では、サービスが割り当てられているグループを調べ、このグループを変更できます。

♦ **サービスが割り当てられているグループを調べて変更するには、次の手順に従います。**

1. コンテキスト・ドロップダウン・リストから [SQL Anywhere 10] プラグインを選択します。
2. 右ウィンドウ枠で、[サービス] タブをクリックします。
3. サービスを選択し、[ファイル] メニューから [プロパティ] を選択します。
4. [依存性] タブをクリックします。最上部のテキスト・ボックスにサービスが割り当てられているグループの名前が表示されます。
5. [変更] をクリックして、システムの使用可能なグループのリストを表示します。
6. いずれかのグループを選択するか、新しいグループの名前を入力します。
7. [OK] をクリックして、そのグループにサービスを割り当てます。

サービス依存性の管理

Sybase Central を使用して、サービスの「**依存性**」を指定できます。次に例を示します。

- サービス・グループのリストにある各グループの少なくとも 1 つのメンバを開始してから現在のサービスを開始するようにできます。
- 任意の数のサービスを開始してから現在のサービスを開始するようにできます。たとえば、特定のネットワーク・サーバを起動してから、そのサーバに対して実行する SQL Remote Message Agent を開始するようにしたい場合などです。

♦ **サービスまたはグループを依存性のリストに追加するには、次の手順に従います。**

1. コンテキスト・ドロップダウン・リストから [SQL Anywhere 10] プラグインを選択します。
2. 右ウィンドウ枠で、[サービス] タブをクリックします。
3. サービスを選択して、[ファイル] – [プロパティ] を選択します。

   [サービス] プロパティ・シートが表示されます。
4. [依存性] タブをクリックします。
5. [サービスの追加] または [サービス グループの追加] をクリックして、サービスまたはグループを依存性リストに追加します。
6. リストからサービスまたはグループを 1 つ選択します。
7. [OK] をクリックして、そのサービスまたはグループを依存性のリストに追加します。

SQL Anywhere の認証アプリケーションの実行

SQL Anywhere の OEM 版は、Sybase OEM パートナに提供されています。SQL Anywhere OEM 版を使用すると、認証アプリケーションは、ユーザ ID に付与されたパーミッションに基づいて、データベース上で任意の操作を実行できます。

非認証で接続すると、読み込み専用アクセスとなり、テンポラリ・テーブルに対する挿入、更新、削除を実行できます。これにより、ストアド・プロシージャを使用して複雑なレポートを作成したり、Crystal Reports などのレポーティング・ツールを使用してアクセスしたりすることができます。

認証メカニズムは、他のすべてのアプリケーション・プログラミング言語やツールとは無関係であり、接続ごとに独立して実行されます。したがって、アプリケーション内で、認証接続と、制限付きの非認証接続の両方を使用できます。

認証は、セキュリティ・メカニズムではありません。データベースに対して非認証データベース・サーバを実行するユーザは、標準的な SQL パーミッション・スキームに基づいて任意の操作を実行できます。

認証アプリケーションの開発

認証アプリケーションの開発は、単純なプロセスです。特殊な認証シグニチャがデータベースに組み込まれ、もう 1 つのシグニチャがアプリケーションに組み込まれます。アプリケーションがデータベースに接続すると、シグニチャが比較され、アプリケーションの認証が行われます。SQL Anywhere の認証アプリケーションを開発するには、以下の手順が必要です。

1. 「認証シグニチャの取得」 49 ページ
2. 「データベースの認証」 49 ページ
3. 「アプリケーションの認証」 50 ページ

SQL Anywhere に付属しているデータベース・ツール (Sybase Central、Interactive SQL、および dbbackup などのユーティリティ) は、自己認証形式です。これは、認証データベースへの操作が制限されないことを意味します。データベース自体が認証されていない場合は、ツールは、制限された読み込み専用モードで動作します。

認証アプリケーションでは、OEM 版の SQL Anywhere データベース・サーバを使用する必要があります。このエディションで、通常のデータベース・サーバと唯一異なる点は、認証命令を処理することです。認証命令は、その他のエディションのデータベース・サーバでは無視されます。認証データベース・サーバを使用しない場合は、非認証アプリケーションにも制限は適用されません。

認証シグニチャの取得

♦ **認証シグニチャを取得するには、次の手順に従います。**

1. Web ブラウザで、http://www.ianywhere.com/downloads/products/sqlanywhere/authentication.html を開きます。
2. フォームに記入し、認証シグニチャを取得します。認証メカニズムに次の情報が組み込まれます。

   ♦ **会社名**　会社の名前。

   ♦ **アプリケーション名**　アプリケーションの名前。

会社名やアプリケーション名が認証メカニズムに組み込まれる方法の詳細については、「データベースの認証」 49 ページを参照してください。

フォームを完成させると、48 時間以内に、データベース・シグニチャとアプリケーション・シグニチャについて知らせる電子メールが送信されます。これらのシグニチャは、英字と数字で構成され、文字数は 81 です。認証情報を含む電子メール・メッセージには、情報の使用法についていくつかの例が記載されています。電子メール・システムの中には、命令の途中で強制的に改行するものがあります。命令を正常に機能させるように、電子メール・メッセージに挿入された改行を削除して行を再結合してください。

データベースの認証

OEM 版の SQL Anywhere では、認証データベースに対する操作は許可されません。

♦ **データベースを認証するには、次の手順に従います。**

1. 次の SQL 認証文を使用して、データベースの database_authentication オプションを設定します。

```
SET OPTION PUBLIC.database_authentication =
    'company = company-name;
    application = application-name;
    signature = database-signature'
```

2. *company-name* と *application-name* の引数は、シグニチャの取得時に Sybase に提供した値です。*database-signature* は、Sybase から受け取ったデータベース・シグニチャです。
3. データベースを再起動して、設定を有効にします。

データベース・サーバが認証データベースをロードするとき、サーバ・メッセージのウィンドウに、認証された会社とアプリケーションについて説明するメッセージが表示されます。このメッセージを調べて、database_authentication オプションが有効になっているかどうかを確認できます。メッセージの形式は次のとおりです。

```
This database is licensed for use with:
Application: application-name
Company: company-name
```


ヒント
SQL スクリプト・ファイルに認証文を格納し、長いシグニチャを繰り返し入力することを回避できます。[ファイル] メニューの [スクリプトの実行] を選択して、Interactive SQL から SQL スクリプトを実行できます。
SQL Anywhere インストール・ディレクトリの *scripts* サブディレクトリに、ファイル *authenticate.sql* を作成し、このファイルに認証文を格納すると、この文は、データベースの作成、再構築、またはアップグレードを行うたびに適用されます。「認証データベースのアップグレード」 52 ページを参照してください。

アプリケーションの認証

認証アプリケーションは、接続確立後すぐに、connection_authentication データベース・オプションを設定する必要があります。このオプションは、接続が確立された直後にすべての接続に対して行われなければなりません。ODBC アプリケーションや JDBC アプリケーションは、データベースに対して機能について問い合わせます。開発者は、これらのアクションを制御する必要はありません。このため、すべての接続には、制限が適用される前に 30 秒間の猶予期間が設けられています。猶予期間を使用して、アプリケーションは、使用している開発ツールには関係なく、認証を行うことができます。

次の SQL 文は、接続を認証します。

```
SET TEMPORARY OPTION connection_authentication =
    'company = company-name;
    application = application-name;
    signature = application-signature'
```

このオプションは、TEMPORARY キーワードを使用して、接続の間だけ設定できます。*company-name* と *application-name* は、データベース認証の文と一致する必要があります。*application-signature* は、Sybase から取得するシグニチャです。

データベース・サーバは、データベース・シグニチャに対してアプリケーション・シグニチャを比較します。シグニチャが確認されると、接続が認証され、SQL パーミッションで設定されている内容にかかわらず、アクティビティは制限されません。シグニチャが確認されないと、接続は、非認証アプリケーションによって許可されているアクションに制限されます。

認証文の実行

認証オプションを設定する SET TEMPORARY OPTION 文を実行する方法は、使用するプログラミング・インタフェースによって異なります。ここに示すシグニチャは、有効なシグニチャではありません。次のインタフェースを使用して認証オプションを設定することを前提に、例を示します。

- ODBC
- PowerBuilder
- JDBC
- ADO.NET
- Embedded SQL

認証オプションにおける特殊文字の使用
会社名に引用符やアポストロフィなどの特殊文字が含まれている場合 (たとえば、Joe's Garage など)、認証文の構成には注意が必要です。認証オプション・セット全体 (Company=...;Application=...;Signature=...) で SQL 文字列になります。SQL における文字列の規則によって、文字列の中に引用符が含まれるかどうかを指定し、受け入れられるようにするには、該当する文字を 2 つ続けて指定する必要があります。次に例を示します。

```
SET TEMPORARY OPTION connection_authentication=
  'Company = Joe''s Garage;
   Application = Joe''s Program;
   Signature = 0fa55157edb8e14d818e...'
```

ODBC

次の文を使用します。

```
SQLExecDirect(
   hstmt,
   "SET TEMPORARY OPTION connection_authentication=
      'Company=MyCo;
       Application=MyApp;
       Signature=0fa55159999e14d818e...'",
   SQL_NTS
);
```

文字列は、1 行に入力します。または、連結して構築します。

PowerBuilder

次の PowerScript 文を使用します。

```
EXECUTE IMMEDIATE
  "SET TEMPORARY OPTION connection_authentication=
     'Company=MyCo;
      Application=MyApp;
      Signature=0fa551599998e14d818e...'"
USING SQLCA
```

JDBC

次の文を使用します。

```
Statement Stmt1 = con.createStatement();
Stmt1.executeUpdate(
   "SET TEMPORARY OPTION connection_authentication=
   'Company=MyCo;
    Application=MyApp;
    Signature=0fa55159999e14d818e...'"
);
```

文字列は、1 行に入力します。または、連結して構築します。

ADO.NET

```
Use the following statement:

SACommand cmd=new SACommand(
  "SET TEMPORARY OPTION connection_authentication=
```

```
        'Company=MyCo;
         Application=MyApp;
         Signature=0fa551599998e14d818e...'",
      con
);
cmd.ExecuteNonQuery();
```

文字列は、1 行に入力します。または、連結して構築します。

Embedded SQL

```
Use the following statement:
EXEC SQL SET TEMPORARY OPTION connection_authentication=
  'Company=MyCo;
   Application=MyApp;
   Signature=0fa551599998e14d818e...';
```

文字列は、1 行に入力します。または、連結して構築します。

認証データベースに接続するとき、接続と認証の手順は別々に行います。ただし、Visual Basic Grid オブジェクトなどの一部のオブジェクトは、独立した暗黙的な接続の試行が可能であり、自動的に認証が行われません。このような場合は、接続は認証されず、データベース操作は失敗します。この問題を回避するには、接続文字列に InitString 接続パラメータを指定します。次の例は、InitString 接続パラメータを含めるように、Visual Basic アプリケーションを修正する例を示したものです。

```
mConnectionString =
  "Provider=SAPROV.10;
   UID=DBA;
   PWD=sql;
   ENG=test10;
   InitString=SET TEMPORARY OPTION connection_authentication=
     'Company=MyCo;
      Application=MyApp;
      Signature=0fa55157edb8e14d818e...'"
mdbName.ConnectionString = mConnectionString
mdbName.Open
mIsSQL = True
```

これにより、すべての接続において、接続確立直後に認証が行われます。

認証データベースのアップグレード

データベースのアップグレードや再構築を行ったときに認証情報を保存するには、ファイル *authenticate.sql* に認証文を格納するしか方法がありません。

アップグレード・ユーティリティはバージョン 10 へのアップグレードに対応していない
アップグレード・ユーティリティ (dbupgrad) では、バージョン 9.0.2 以前のデータベースをバージョン 10 にアップグレードすることはできません。以前のバージョンのデータベースをバージョン 10 にアップグレードするには、アンロードと再ロードを実行し、データベースを再構築する必要があります。「SQL Anywhere のアップグレード」『SQL Anywhere 10 - 変更点とアップグレード』を参照してください。

次の内容の *authenticate.sql* というファイルを *install-dir¥scripts* ディレクトリに作成します。

```
SET OPTION PUBLIC.database_authentication = 'authentication-statement'
   go
```

ファイルには go が含まれている必要があります。含まれていないと、この文は無視されます。

authentication-statement 文字列の内容については、「database_authentication [データベース]」 440 ページを参照してください。

サーバ起動時のトラブルシューティング

この項では、データベース・サーバの起動時に発生する一般的な問題について説明します。

トランザクション・ログ・ファイルが有効であるかを確認する

既存のトランザクション・ログが無効だと、サーバは起動しません。たとえば、開発中に、データベース・ファイルを新しいバージョンに置き換え、同時にトランザクション・ログを削除しなかったとします。これは、トランザクション・ログとデータベースが異なる原因となり、結果として無効なトランザクション・ログになります。

テンポラリ・ファイル用の十分なディスク領域があるかを確認する

SQL Anywhere は、テンポラリ・ファイルを使用して実行時に情報を格納します。このファイルは、SATMP 環境変数で指定されるディレクトリ (通常は *c:¥temp*) に保存されます。

テンポラリ・ディレクトリに使用できる十分なディスク領域がないと、サーバを起動するときに問題が生じることがあります。

「SATMP 環境変数」 314 ページを参照してください。

ネットワーク通信ソフトウェアが実行されているかを確認する

適切なネットワーク通信ソフトウェアをインストールして実行してから、データベース・サーバを実行します。信頼性の高いネットワーク・ソフトウェアを 1 つのネットワークだけが導入された状態で実行している場合は、問題はありません。

ネットワーク通信問題に関する詳細については、「クライアント／サーバ通信」 121 ページを参照してください。

データベース・サーバを実行する前に、ネットワーク通信を必要とする他のソフトウェアが正しく動作していることを確認してください。

TCP/IP プロトコルを使用している場合は、ping と Telnet が正しく動作していることを確認します。ping アプリケーションと Telnet アプリケーションは、多数の TCP/IP プロトコル・スタックを提供します。

ネットワーク通信の起動時の問題をデバッグする

ネットワークで接続を確立するときに問題が生じた場合は、クライアントとサーバの両方でデバッグ・オプションを使用すると、問題点を診断できます。サーバ側で、-z オプションを使用します。サーバ・メッセージ・ウィンドウに起動情報が表示されます。-o オプションを使用して、出力ファイルに結果のログを取ります。

「-z サーバ・オプション」 210 ページと「-o サーバ・オプション」 182 ページを参照してください。

正しい sasrv.ini ファイルを使用していることの確認

ネットワーク経由で正しいサーバへの接続を確立することに問題がある場合、*sasrv.ini* ファイルを削除してみてください。このファイルには、サーバ名、プロトコル、アドレスなどのサーバ情報が入っています。このファイルにあるサーバ情報が、接続文字列に指定した情報を上書きしている可能性があります。このファイルを削除すると、SQL Anywhere は、ユーザが接続文字列に指定した情報を含む新しい *sasrv.ini* ファイルを作成します。*sasrv.ini* のデフォルトのロケーションは、Windows では *%ALLUSERSPROFILE%¥Application Data¥SQL Anywhere 10*、UNIX では *~/.sqlanywhere10* です。

それでもまだ接続を確立できないときは、次のいずれかのロケーションに *sasrv.ini* があれば、このファイルをすべて削除します。

- SQL Anywhere インストール・ディレクトリの *win32*、*ia64*、または *x64* サブディレクトリ (*HKEY_LOCAL_MACHINE¥SOFTWARE¥Sybase¥SQL Anywhere¥10.0¥Location* レジストリ・キーにリストされるディレクトリに含まれます)
- Windows ディレクトリ
- Windows システム・ディレクトリ
- パスに指定されたいずれかのロケーション

sasrv.ini ファイルの詳細については、「迅速な接続のためのサーバ名キャッシュ」 95 ページを参照してください。

デバッグ・ログ・ファイルの作成

LogFile 接続パラメータを使用してデバッグ・ログ・ファイルを作成できます。ログ・ファイルは、接続障害が発生した場所についてより詳細な情報を提供できるので、問題のトラブルシューティングや修正に役立ちます。「LogFile 接続パラメータ [LOG]」 256 ページを参照してください。

SQL Anywhere のエラー・レポート

致命的なエラーやクラッシュが発生し、次に挙げるアプリケーションで検出されると、問題が発生したときの状況に関するエラー・レポートが作成されます。

- Interactive SQL (dbisql)
- Mobile Link Listener (dblsn)
- Mobile Link サーバ (mlsrv)
- ネットワーク・サーバ (dbsrv10)
- パーソナル・サーバ (dbeng10)
- QAnywhere Agent (qaagent)
- Replication Agent (dbltm)
- Mobile Link 用 SQL Anywhere クライアント (dbmlsync)
- SQL Anywhere コンソール・ユーティリティ (dbconsole)
- SQL Remote (dbremote)
- Sybase Central

エラー・レポートには、クラッシュ発生時のスレッドの実行状況などの情報が含まれます。これによって、iAnywhere では、より的確に問題の原因を究明できます。デフォルトでは、エラー・レポートは診断ディレクトリに保存されます (このディレクトリは、SADIAGDIR 環境変数で指定されています)。このディレクトリが存在しない場合は、データベース・ファイルと同じディレクトリに作成されます。エラー・レポート・ファイルの場所は、コンソール・ログと要求ログに書き込まれます。

エラー・レポートのファイル名は、次のように構成されます。

- アプリケーションを識別するプレフィクス

アプリケーションの識別子	アプリケーション
LSN	Listener ユーティリティ
LTM	Replication Agent
MLC	Mobile Link クライアント
MLS	Mobile Link サーバ
QAA	QAnywhere Agent
SA	パーソナル・データベース・サーバまたはネットワーク・データベース・サーバ
SR	SQL Remote

- ソフトウェアのバージョンを示す数値
- アンダースコア付きでリンクが設定された 2 つのフィールド (エラー・レポートが作成されたときのタイムスタンプを提供)

- アプリケーションの識別子
- 拡張子 *.mini_core*

たとえば、*SA10_20051220_133828_32116.mini_core* は、SQL Anywhere バージョン 10 のデータベース・サーバで、2005/12/20 の午後 1:38:28 に、プロセス 32116 から作成されたエラー・レポートであることを示します。

サーバの正常稼働中においても、コンピュータ上の CPU の数、ハイパースレッドが有効かどうか、サーバの起動時に指定されたオプションなど、データベース・サーバについてのさまざまな診断情報が記録されています。この情報は、dbsupport を使用して送信することもできます。

SQL Anywhere ソフトウェアがエラー・レポートと診断情報を送信する方法

エラー・レポート情報の出力が完了すると、データベース・サーバは SQL Anywhere サポート・ユーティリティ (dbsupport) を起動し、送信するエラー・レポート・ファイルの名前を渡します。デフォルトでは、エラー・レポートが生成されると、送信するかどうかを確認するメッセージが表示されます。このメッセージを表示できない場合、エラー・レポートは送信されません。メッセージが表示されるときには、iAnywhere によって、エラー・レポートを送信するように選択することをすすめられます。レポートには、送信者が特定される情報は含まれません。

この dbsupport のデフォルト動作を変更するには、-cc オプションを使用します。

- 次のコマンドは、ユーザにメッセージを表示せず、自動的にエラー・レポートを送信するように dbsupport を設定します。

  ```
  dbsupport -cc autosubmit
  ```

- 次のコマンドは、自動的なエラー・レポート送信を無効にします。

  ```
  dbsupport -cc no
  ```

エラー・レポートを送信しないように選択すると、レポートの内容はハード・ディスクの診断ディレクトリに維持されます。診断ディレクトリの場所は、SADIAGDIR 環境変数で指定されています。「SADIAGDIR 環境変数」 311 ページを参照してください。

エラー・レポートを iAnywhere に送信すると、致命的なエラーやアサーションの原因の究明に役立つことがあります。送信したエラー・レポートは、コンピュータから削除されます。「SQL Anywhere サポート・ユーティリティ (dbsupport)」 722 ページを参照してください。

エラー・レポートと診断情報は、HTTP 経由で iAnywhere の Error Reporting Web サイトにアップロードされます。このように問題の診断と解決策の提供に役立つファイルを簡単に iAnywhere へ送信できるため、ユーザにとっては時間の節約になります。

第 3 章

データベースへの接続

目次

SQL Anywhere データベース接続の概要

データベースを使用するクライアント・アプリケーションでは、なんらかの作業を行う前に必ずそのデータベースへの「**接続**」を確立する必要があります。接続は、チャネルを形成します。クライアント・アプリケーションからのアクティビティは、すべてそのチャネルを介して行われます。たとえば、データベースではユーザ ID に応じてアクションを実行するパーミッションが決定しますが、データベース・サーバがこのユーザ ID を認識するのは、ユーザ ID が接続確立要求の一部として送信されるからです。

ユーザがデータベースに接続する場合、データベース・サーバはその接続にユニークな「**接続 ID**」を割り当てます。データベース・サーバに対して新たに接続するたびに、サーバは接続 ID 値を 1 つずつ増やします。これらの接続 ID は、-z サーバ出力によってログされます。接続 ID は、要求ログ情報のフィルタリング、データベースにロックがある接続の識別、起動してからのサーバへの合計接続数やその接続の順番の追跡に使用できます。

「要求ロギング」『SQL Anywhere サーバ - SQL の使用法』と「ロックの仕組み」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

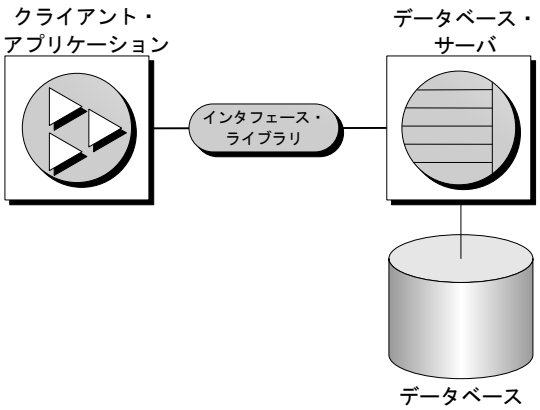
SQL Anywhere API からの接続

接続を確立するために、クライアント・アプリケーションは SQL Anywhere インタフェースのいずれかで関数を呼び出します。SQL Anywhere が提供するインタフェースを次に示します。

インタフェース	詳細
ODBC	「SQL Anywhere ODBC API」『SQL Anywhere サーバ - プログラミング』 「ODBC データ・ソースの使用」 75 ページ
OLE DB	「OLE DB を使用したデータベースへの接続」 84 ページ
ADO.NET	「データベースへの接続」『SQL Anywhere サーバ - プログラミング』
Embedded SQL	「SQL Anywhere Embedded SQL」『SQL Anywhere サーバ - プログラミング』
Sybase Open Client	「Open Server としての SQL Anywhere」 985 ページ 「Sybase Open Client API」『SQL Anywhere サーバ - プログラミング』
iAnywhere JDBC ドライバ	「JDBC クライアント・アプリケーションからの接続」『SQL Anywhere サーバ - プログラミング』 「SQL Anywhere JDBC API」『SQL Anywhere サーバ - プログラミング』

インタフェース	詳細
jConnect JDBC ドライバ	「JDBC クライアント・アプリケーションからの接続」 『SQL Anywhere サーバ - プログラミング』 「SQL Anywhere JDBC API」 『SQL Anywhere サーバ - プログラミング』

インタフェースは、クライアント・アプリケーションからの呼び出しに含まれている接続情報を使用して、要求されたデータベースを実行しているサーバを見つけて接続します。このとき、データ・ソース、SQLCONNECT 環境変数、またはサーバ・アドレス・キャッシュに格納されている情報を併せて使用することもあります。次の図に、関連する各部分を簡単に示します。



参照先

次の表に、各質問とその回答が記述されている参照先を示します。

目的	参照先
Sybase Central または Interactive SQL からの接続の概要 (関係するドライバの説明も含む)	「Sybase Central、Interactive SQL、SQL Anywhere コンソール・ユーティリティからの接続」 64 ページ
高速起動の例 (Sybase Central と Interactive SQL のシナリオも含む)	「簡単な接続の例」 66 ページ
データ・ソースについて知りたい	「ODBC データ・ソースの使用」 75 ページ
使用できる接続パラメータについて知りたい	「接続パラメータ」 230 ページ
接続の確立方法について詳しく知りたい	「接続のトラブルシューティング」 88 ページ
ネットワーク固有の接続問題について知りたい	「クライアント／サーバ通信」 121 ページ
接続に影響する文字セット問題について知りたい	「接続文字列と文字セット」 344 ページ

目的	参照先
ファイアウォールを経由した接続について知りたい	「ファイアウォール経由の接続」 124 ページ

接続パラメータの働き

アプリケーションはデータベースに接続するとき、一連の「**接続パラメータ**」を使用して接続を定義します。接続パラメータには、サーバ名、データベース名、ユーザ ID などの情報が含まれています。

キーワードと値の組み合わせ (形式は *parameter=value*) で、各接続パラメータを指定します。たとえば、デフォルト・パスワードのパスワード接続パラメータは、次のように指定します。

```
Password=sql
```

接続パラメータは、アセンブルされて「**接続文字列**」になります。接続文字列では、各接続パラメータをセミコロンで区切ります。例を次に示します。

```
ServerName=demo10;DatabaseName=demo
```

接続文字列の表現

この章では、次の形式で表現される接続文字列の例を多数紹介しています。

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

この接続文字列は、次の接続文字列と同じ意味を持ちます。

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

接続文字列は、各パラメータ設定をセミコロンで区切り、1 行で入力してください。

接続文字列として渡される接続パラメータ

接続パラメータは、「**接続文字列**」としてインタフェース・ライブラリに渡されます。この文字列は、次のようなセミコロンで区切った一連のパラメータから成ります。

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

通常、アプリケーションが構築してインタフェース・ライブラリに渡す接続文字列は、ユーザが情報を入力する方法と直接は対応していません。代わりに、ユーザがダイアログ・ボックスに入力するか、アプリケーションが初期化ファイルから接続情報を読み込みます。

SQL Anywhere のユーティリティの多くは、接続文字列を -c オプションとして認識し、変更を加えないでインタフェース・ライブラリに渡します。たとえば、典型的なバックアップ・ユーティリティ (dbbackup) コマンド・ラインを次に示します。

```
dbbackup -c "ENG=sample_server;DBN=demo;UID=DBA;PWD=sql" SQLAnybackup
```

参照

- 「接続パラメータのヒント」 86 ページ

ODBC データ・ソースでの接続パラメータの保存

多くのクライアント・アプリケーションは、アプリケーション開発システムも含め、ODBC インタフェースを使用して SQL Anywhere にアクセスします。データベースに接続するときは、ODBC アプリケーションは ODBC データ・ソースを使用するのが一般的です。ODBC データ・ソースは一連の接続パラメータで、レジストリまたはファイルに格納されています。

SQL Anywhere の場合は、Open Client と jConnect 以外のすべてのクライアント・インタフェースで ODBC データ・ソースを使用できます。UNIX と Windows CE では、データ・ソースはファイルに保管されます。ODBC データ・ソースは NetWare では使用できません。

参照

- 「ODBC データ・ソースの使用」 75 ページ

Sybase Central、Interactive SQL、SQL Anywhere コンソール・ユーティリティからの接続

Sybase Central、Interactive SQL、SQL Anywhere コンソール・ユーティリティを使用してデータベースを管理するには、まず接続します。[接続] ダイアログで、接続するデータベース、データベースの場所、接続方法を Sybase Central、Interactive SQL、SQL Anywhere コンソールに指定します。

接続処理は状況によって異なります。たとえば、ユーザが自分のコンピュータでサーバを実行中で、このサーバにはデータベースが 1 つしかない場合、[接続] ダイアログではそのデータベースのユーザ ID とパスワードだけを入力します。すると、Sybase Central、Interactive SQL、SQL Anywhere コンソールが、実行中のサーバのデータベースにすぐに接続します。

この実行中のサーバにロード済みのデータベースが複数ある場合、サーバがまだ実行されていない場合、または別のコンピュータでサーバが実行中の場合は、[接続] ダイアログで、さらに詳しい情報を入力する必要があります。Sybase Central、Interactive SQL、SQL Anywhere コンソールは、この情報を元に接続するデータベースを識別します。

参照

- 「簡単な接続の例」 66 ページ

[接続] ダイアログを開く

Sybase Central と Interactive SQL では、共通の [接続] ダイアログを使用してデータベースに接続できます。

Sybase Central の起動時は、このダイアログを手動で開く必要があります。Interactive SQL を起動した場合、ダイアログは自動的に表示されます。[SQL] – [接続] を選択して、新しい接続についてのダイアログを表示することもできます。

- **[接続] ダイアログを開くには、次の手順に従います (Sybase Central の場合)。**

1. Sybase Central で、[接続] – [SQL Anywhere 10 に接続] を選択します。

   [F11] キーを押すと、[接続] メニューが開きます。

> **ヒント**
> **接続プロファイル**を使用すると、指定されたデータベースへの 2 回目以降の接続を簡単かつ迅速に行えます。

- **[接続] ダイアログを表示するには、次の手順に従います (Interactive SQL の場合)。**

1. Interactive SQL から、[SQL] – [接続] を選択します。

   別の方法として、[F11] を押して [接続] ダイアログを開きます。

[接続] ダイアログが表示されたら、接続に必要な接続パラメータを指定します。たとえば、サンプル・データベースに接続するには、[ID] タブにある [ODBC データ・ソース名] のリストから [SQL Anywhere 10 Demo] を選択し、[OK] をクリックします。

♦ [接続] ダイアログを表示するには、次の手順に従います (SQL Anywhere コンソール・ユーティリティの場合)。

· コマンド・プロンプトで dbconsole と入力します。

[接続] ダイアログ・ボックスが表示されます。

[接続] ダイアログの使用

[接続] ダイアログでは、サーバまたはデータベースへの接続で使用するパラメータを定義できます。Sybase Central でも Interactive SQL でも同じダイアログが使用されます。[接続] ダイアログに入力した情報は、セッション間では保持されません。

[接続] ダイアログには、次のタブがあります。

- [ID] タブ。データベースにユーザが誰かを識別させ、またここでデータ・ソースを指定します。
- [データベース] タブ。接続するサーバかデータベースまたはその両方を特定します。
- [詳細] タブ。追加接続パラメータを追加し、その接続に使用するドライバを指定します。

Sybase Central では、接続が成功すると、実行中のサーバ上でメイン・ウィンドウの左ウィンドウ枠にデータベース名が表示されます。データベース名の後ろにこの接続のユーザ ID が表示されます。

Interactive SQL では、接続情報 (データベース名、ユーザ ID、データベース・サーバなど) は、[SQL 文] ウィンドウ枠の上のタイトル・バーに表示されます。

簡単な接続の例

SQL Anywhere では接続モデルを設定でき、複雑にすることもできますが、多くの場合、データベースへの接続は非常に簡単です。

この項の内容

この項では、アプリケーションが SQL Anywhere データベースに接続する簡単な事例について説明します。この項の情報は、接続を開始するすべてのユーザを対象としています。

参照

- 「接続パラメータ」 230 ページ

Sybase Central または Interactive SQL からのサンプル・データベースへの接続

このマニュアルでは、多くの例で最初に Sybase Central または Interactive SQL からサンプル・データベースに接続しています。

- **サンプル・データベースに接続するには、次の手順に従います (Sybase Central の場合)。**

1. [スタート] メニューから、[プログラム] − [SQL Anywhere 10] − [Sybase Central] を選択して、Sybase Central を起動します。
2. [接続] ダイアログを開きます。[接続] − [SQL Anywhere 10 に接続] を選択します。
3. [ODBC データ・ソース名] オプションを選択し、[参照] をクリックします。
4. [SQL Anywhere 10 Demo] を選択して、[OK] をクリックします。

- **サンプル・データベースに接続するには、次の手順に従います (Interactive SQL の場合)。**

1. [スタート] メニューから、[プログラム] − [SQL Anywhere 10] − [Interactive SQL] を選択して、Interactive SQL を起動します。
2. [SQL] メニューから [接続] を選択して、[接続] ダイアログを開きます。
3. [ODBC データ・ソース名] オプションを選択し、[参照] をクリックします。
4. [SQL Anywhere 10 Demo] を選択して、[OK] をクリックします。

   [SQL 文] ウィンドウ枠の上のタイトル・バーに、データベース名、ユーザ ID、サーバ名が表示されます。

注意
この接続では、ユーザ ID とパスワードはすでにデータ・ソースにあるので入力する必要はありません。

♦ **サンプル・データベースに接続するには、次の手順に従います (データベース・ファイルのロケーションを指定する場合)**

1. Sybase Central または Interactive SQL で、[接続] ダイアログを開きます。
2. [ID] タブで、次の値を入力します。
   - ♦ **[ユーザ ID]** **DBA** と入力します。
   - ♦ **[パスワード]** **sql** と入力します。
3. [データベース] タブで、サンプル・データベース *demo.db* の名前を入力するか、このファイルのロケーションを参照します。デフォルトで、このファイルは *samples-dir* にあります。たとえば、Windows XP の場合、デフォルト・ロケーションは *C:¥Documents and Settings¥All Users¥Documents¥SQL Anywhere 10¥Samples¥demo.db* です。

   samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。
4. [OK] をクリックします。

Sybase Central または Interactive SQL から同じコンピュータにあるデータベースへの接続

接続シナリオが最も簡単になるのは、接続したいデータベースが同じコンピュータにあるときです。このような場合があるときは、次の質問を読んでください。

- ♦ データベースをすでにサーバで実行しているか。そうでない場合は、Sybase Central または Interactive SQL がデータベースを起動できるように、データベース・ファイルを特定する必要があります。実行している場合、[接続] ダイアログでパラメータを減らせます。
- ♦ 使用しているコンピュータでデータベースを複数実行しているか。1 つだけ実行している場合は、Sybase Central または Interactive SQL がそのデータベースを接続するものと判断するので、[接続] ダイアログで指定する必要はありません。複数実行している場合は、接続するデータベースを特定するために Sybase Central または Interactive SQL に指示を与える必要があります。

次の手順は、上記の質問に対する回答によって異なります。

♦ **すでに実行中のローカル・サーバ上でデータベースに接続するには、次の手順に従います。**

1. Sybase Central または Interactive SQL を起動します。[接続] ダイアログが自動的に表示されなかった場合、[接続] ダイアログを開きます。
2. ダイアログの [ID] タブで、現在実行中のデータベースのユーザ ID とパスワードを入力します。
3. 次のいずれかを行います。

- サーバで実行しているデータベースが 1 つしかない場合は、[OK] をクリックしてそのサーバに接続します。
- サーバで実行しているデータベースが複数ある場合は、ダイアログの [データベース] タブをクリックし、データベース名を指定します。これは通常はデータベース・ファイル名で、パスや拡張子は付けません。

♦ **データベースを起動して接続するには、次の手順に従います。**

1. Sybase Central または Interactive SQL を起動します。[接続] ダイアログが自動的に表示されなかった場合、[接続] ダイアログを開きます。
2. ダイアログの [ID] タブで、ユーザ ID とパスワードを入力します。
3. [データベース] タブをクリックします。
4. [データベース・ファイル] フィールドでファイルを指定します。ここでは、フル・パス、名前、拡張子を含めます。ファイルを検索するには、[参照] をクリックします。
5. 次回接続するときのデータベース名を現在のファイル名とは別の名前にしたい場合は、[データベース名] フィールドに名前を入力します (パスや拡張子は含めません)。

ヒント
データベースがサーバにロード (起動) されている場合は、データベース名を指定するだけで接続が成功します。データベース・ファイルは必要ありません。
接続には、データ・ソース (格納されている一連の接続パラメータ) を上記のいずれのシナリオにも使用できます。[接続] ダイアログの [ID] タブの下部で適切なデータ・ソース・オプションを選択します。

参照

- 「簡単な接続の例」 66 ページ

組み込みデータベースへの接続

「**組み込みデータベース**」は、単一アプリケーションで使用するために設計されており、アプリケーションと同じコンピュータ上で稼働し、その大部分はアプリケーション・ユーザからは隠されています。

アプリケーションが組み込みデータベースを使用する場合、アプリケーションの接続時には、通常、パーソナル・サーバは動作していません。この場合、データベースの開始には接続文字列を使用し、接続文字列の DatabaseFile (DBF) パラメータにデータベース・ファイルを指定します。

DBF パラメータの使用

DatabaseFile (DBF) パラメータは、使用するデータベース・ファイルを指定します。データベース・ファイルはデフォルトのサーバに自動的にロードされます。または、実行中のサーバがない場合はサーバが起動されます。

データベースへの接続がなくなったら (通常は接続を開始したアプリケーションが切断したら)、データベースはアンロードします。接続によりサーバが起動された場合は、データベースのアンロードと同時にデータベース・サーバは停止されます。

次の接続パラメータは、サンプル・データベースを組み込みデータベースとしてロードする方法を示しています。

```
DBF=samples-dir¥demo.db
UID=DBA
PWD=sql
```

samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

ENG パラメータの使用

組み込みデータベースを使用する場合は、EngineName (ENG) 接続パラメータを使用することをおすすめします。これにより、同じコンピュータ上で SQL Anywhere データベース・サーバを実行している別のアプリケーションが存在する場合でも、データベースが適切なデータベース・サーバに接続されます。

StartLine [START] パラメータの使用

次の接続パラメータは、組み込みデータベースとしてのサンプル・データベースの開始をカスタマイズする方法を示しています。これは、キャッシュ・サイズなどのオプションを使用する場合に便利です。

```
START=dbeng10 -c 8M
DBF=samples-dir¥demo.db
UID=DBA
PWD=sql
```

サーバの起動方法に影響する接続パラメータが多数存在します。StartLine (START) 接続パラメータで対応するサーバ・オプションを使用するのではなく、次の接続パラメータを使用することをおすすめします。

- EngineName (ENG)
- DatabaseFile (DBF)
- DatabaseSwitches (DBS)
- DatabaseName (DBN)

参照

- 「[接続] ダイアログを開く」 64 ページ
- 「簡単な接続の例」 66 ページ

データ・ソースを使用した接続

一連の接続パラメータを「**データ・ソース**」に保存できます。Open Client と jConnect 以外のすべての SQL Anywhere インタフェースでは、データ・ソースを使用できます。

♦ データ・ソースを使用して接続するには、次の手順に従います (Sybase Central または Interactive SQL の場合)。

1. Sybase Central または Interactive SQL を起動します。[接続] ダイアログが自動的に表示されなかった場合、[接続] ダイアログを開きます。
2. [ID] タブで、ユーザ ID とパスワードを入力します。
3. [ID] タブの下部で、次のいずれかを指定します。
   - [ODBC データ・ソース名] オプションを選択し、データ・ソース名を指定します。データ・ソース名は、Windows レジストリ内でデータ・ソースを参照する DataSourceName (DSN) 接続パラメータと同じになります。データ・ソースのリストを表示するには、[参照] をクリックします。
   - [ODBC データ・ソース・ファイル] オプションを選択し、データ・ソース・ファイルを指定します。データ・ソース・ファイルは、ファイルにあるデータ・ソースを参照する FileDataSourceName (FILEDSN) 接続パラメータと同じになります。ファイルを検索するには、[参照] をクリックします。

SQL Anywhere 10 Demo という名前のデータ・ソースには、データベース・ファイルや、データベースを開始する StartLine (START) パラメータなど、一連の接続パラメータが格納されています。

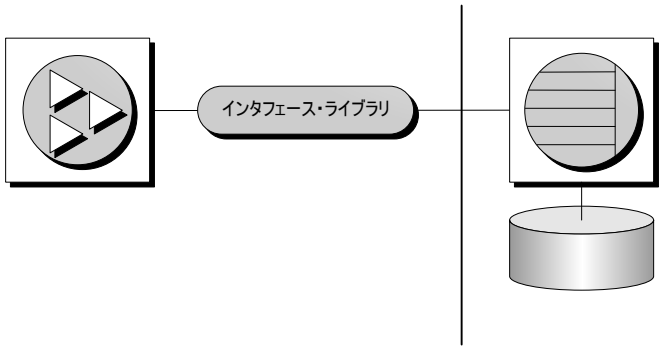
参照

- 「[接続] ダイアログを開く」 64 ページ
- 「簡単な接続の例」 66 ページ
- 「UNIX での ODBC データ・ソースの使用」 81 ページ

ネットワーク上のサーバへの接続

LAN または WAN のどこかにあるネットワーク・サーバ上で動作するデータベースに接続するには、クライアント・ソフトウェアがそのデータベース・サーバを見つけ、それに接続します。SQL Anywhere は、ネットワーク・ライブラリを提供してこのタスクを処理します。

ネットワーク接続は、「**ネットワーク・プロトコル**」を介して行います。TCP/IP はすべてのプラットフォームで、SPX は一部のプラットフォームで使用できます。



サーバの指定

SQL Anywhere のサーバ名は、所定のネットワーク・プロトコルのローカル・ドメイン上ではユニークである必要があります。次の接続パラメータは、ネットワーク上のどこか別の場所で稼働しているサーバに接続する場合の簡単な例を示します。

```
ENG=svr_name
DBN=db_name
UID=user_id
PWD=password
CommLinks=all
```

CommLinks=all を指定すると、クライアント・ライブラリは、指定された名前のパーソナル・サーバを最初に探索し、次に指定された名前のサーバをネットワークで探索します。「CommLinks 接続パラメータ [LINKS]」 235 ページを参照してください。

プロトコルの指定

複数のプロトコルを使用できる場合は、パフォーマンスの改善のために使用するプロトコルをネットワーク・ライブラリに対して指定できます。次のパラメータは、TCP/IP プロトコルのみを使用します。

```
ENG=svr_name
DBN=db_name
UID=user_id
PWD=password
CommLinks=tcpip
```

ネットワーク・ライブラリは、ネットワークをブロードキャストしてサーバを検索します。この処理は時間がかかることがあります。ネットワーク・ライブラリがサーバを見つけると、クライアント・ライブラリは、そのサーバの名前とネットワーク・アドレスをファイル (*sasrv.ini*) に保存します。以降の接続にはこのエントリを再利用し、指定されたプロトコルを使用してサーバに接続しようとします。以降の接続は、通常ブロードキャストによって確立された接続よりも速くなります。

この他にも、ネットワーク上で効率的にサーバを検索するときに、SQL Anywhere を支援できる接続パラメータがたくさんあります。

♦ ネットワーク・サーバ上のデータベースに接続するには、次の手順に従います (Sybase Central または Interactive SQL の場合)。

1. Sybase Central または Interactive SQL を起動します。[接続] ダイアログが自動的に表示されなかった場合、[接続] ダイアログを開きます。
2. ダイアログの [ID] タブで、ユーザ ID とパスワードを入力します。
3. ダイアログの [データベース] タブの [サーバ名] に値を入力します。[ネットワーク上でデータベース・サーバを検索] オプションを選択すると、サーバを検索できます。
4. [データベース名] を指定してデータベースを識別します。

ヒント
[接続] ダイアログの [ID] タブの一番下にある適切なデータ・ソース・オプションを選択すると、データ・ソース (保存されている接続パラメータ・セット) を使用して接続できます。デフォルトでは、Sybase Central と Interactive SQL のすべてのネットワーク接続は、TCP/IP ネットワーク・プロトコルを使用するようになっています。

参照

- 「クライアント／サーバ通信」 121 ページ
- 「ネットワーク・プロトコル・オプション」 265 ページ
- 「ファイアウォール経由の接続」 124 ページ
- 「[接続] ダイアログを開く」 64 ページ
- 「簡単な接続の例」 66 ページ

デフォルト接続パラメータの使用

接続パラメータの多くに何も指定しないで、デフォルトの動作を代わりに使用して接続することもできます。ただし、運用環境でデフォルトの動作に依存する場合は注意してください。特に、コンピュータに他の SQL Anywhere アプリケーションをインストールしている可能性がある顧客にアプリケーションを配布する場合は注意が必要です。

デフォルト動作の詳細については、「接続のトラブルシューティング」 88 ページを参照してください。

デフォルトのデータベース・サーバとデータベース

パーソナル・サーバが 1 つだけ実行されていて、そこにデータベースが 1 つだけロードされている場合は、デフォルトのパラメータをすべて使用して接続できます。

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

デフォルトのデータベース・サーバ

1 つのパーソナル・サーバに複数のデータベースがロードされている場合は、サーバはデフォルトのままにしておくこともできますが、接続したいデータベースを指定する必要があります。

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

デフォルトのデータベース

複数のサーバが動作している場合は、接続するサーバを指定してください。そのサーバにデータベースが 1 つしかロードされていない場合は、データベース名を指定する必要はありません。次の接続文字列は、指定されたサーバに接続して、デフォルト・データベースを使用します。

```
ENG=server-name
UID=user-id
PWD=password
```

デフォルトを使用しない場合

次の接続文字列は、指定されたローカル・サーバに接続して、指定されたデータベースを使用します。

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

デフォルトを使用しない場合

異なるコンピュータで実行中のネットワーク・サーバに接続するには、次の手順に従います。

```
ENG=server-name
DBN=dbn
UID=user-id
PWD=password
CommLinks=tcpip
```

CommLinks が指定されていない場合は、ローカルの共有メモリ接続のみが行われます。

Sybase Central、Interactive SQL、または SQL Anywhere コンソール・ユーティリティ (dbconsole) から接続している場合、[接続] ダイアログの [ネットワーク上でデータベース・サーバを検索] オプションを選択して、ネットワーク接続を行うことができます。

SQL Anywhere ユーティリティからの接続

サーバと通信するすべての SQL Anywhere データベース・ユーティリティは、データベース・ファイルを直接処理するのではなく、Embedded SQL を使用して処理します。

データベースに接続するときにユーティリティが使用する手順の詳細については、「接続のトラブルシューティング」 88 ページを参照してください。

データベース・ユーティリティが接続パラメータの値を取得する方法

多くの管理ユーティリティは、接続パラメータの値を次の方法で取得します。

1. コマンド・ラインで指定された値があれば、それを使用します。たとえば、次のコマンドは、ユーザ ID DBA とパスワード sql を使用して、デフォルト・サーバ上のデフォルト・データベースのバックアップを開始します。

   ```
   dbbackup -c "UID=DBA;PWD=sql" c:¥backup
   ```


各データベース・ユーティリティのオプションの詳細については、「データベース管理ユーティリティ」 635 ページを参照してください。

2. 値がない場合は、SQLCONNECT 環境変数の設定を使用します。SQL Anywhere では、この変数は自動的に設定されるわけではありません。

   「SQLCONNECT 環境変数」 318 ページを参照してください。

ODBC データ・ソースの使用

Microsoft では、「**オープン・データベース・コネクティビティ (ODBC)**」インタフェースを、Windows 環境下でクライアント・アプリケーションをデータベース管理システムに接続する標準のインタフェースとして定義しています。アプリケーション開発システムなど、多くのクライアント・アプリケーションは、ODBC インタフェースを使用して、さまざまなデータベース・システムにアクセスします。

SQL Anywhere ODBC ドライバの名前は *dbodbc10.dll* で、*install-dir¥win32* にあります。

SQL Anywhere での ODBC の使用方法の詳細については、「ODBC 準拠」『SQL Anywhere サーバ - プログラミング』を参照してください。

ODBC データ・ソースを使用して SQL Anywhere データベースに接続できるアプリケーションには、次のものがあります。

- Sybase Central と Interactive SQL
- すべての SQL Anywhere ユーティリティ
- PowerDesigner Physical Data Model と InfoMaker
- Microsoft Visual Basic、Sybase PowerBuilder、Borland Delphi など、ODBC をサポートしているアプリケーション開発環境
- UNIX の SQL Anywhere クライアント・アプリケーション UNIX では、データ・ソースはファイルとして保存されます。

データ・ソースの格納場所

ODBC を使用してデータベースに接続するには、ODBC データ・ソースを使用します。クライアント・コンピュータ上には、接続するデータベースごとに ODBC データ・ソースが必要です。

ODBC データ・ソースには、一連の接続パラメータが格納されています。SQL Anywhere の一連の接続パラメータは、ODBC データ・ソースとして、Windows レジストリに格納するか、ファイルとして格納できます。

データ・ソースがあれば、接続文字列にはデータ・ソースを次のように指定するだけです。

- **データ・ソース**　DataSourceName (DSN) 接続パラメータを使用して、Windows レジストリ内のデータ・ソースを参照する。

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

- **ファイル・データ・ソース**　FileDataSourceName (FILEDSN) 接続パラメータを使用して、ファイルに格納されているデータ・ソースを参照する。

  ```
  FileDSN=mysource.dsn
  ```

SQL Anywhere では、ODBC インタフェースを使用する Windows アプリケーション以外でも ODBC データ・ソースを使用できます。

- UNIX 上の SQL Anywhere クライアント・アプリケーションは、Windows オペレーティング・システム上のものと同じように、ODBC データ・ソースを使用できます。
- ODBC データ・ソースは、jConnect と Open Client 以外のすべての SQL Anywhere クライアント・インタフェースで使用できます。

注意
作成する接続文字列には、明示的に指定された接続パラメータの他に、接続パラメータを含む ODBC データ・ソースの名前を含めることができます。接続パラメータが接続文字列に ODBC データ・ソースとともに指定された場合、明示的に指定された値が優先されます。

ODBC データ・ソースの作成

ODBC データ・ソースを Windows オペレーティング・システムで作成するには、ODBC アドミニストレータを使用します。これは、ODBC データ・ソースの作成と管理の中枢を成します。

SQL Anywhere には、データ・ソースを作成する dbdsn というプラットフォームを問わないユーティリティも含まれています。

ODBC アドミニストレータを使用した ODBC データ・ソースの作成

Windows では、Microsoft ODBC アドミニストレータを使用してデータ・ソースの作成と編集ができます。このユーティリティでは、ユーザ・データ・ソース、ファイル・データ・ソース、システム・データ・ソースについて処理できます。

♦ ODBC データ・ソースを作成するには、次の手順に従います (ODBC アドミニストレータの場合)。

1. [スタート] メニューから、[プログラム] – [SQL Anywhere 10] – [SQL Anywhere] – [ODBC アドミニストレータ] の順に選択して、ODBC アドミニストレータを起動します。

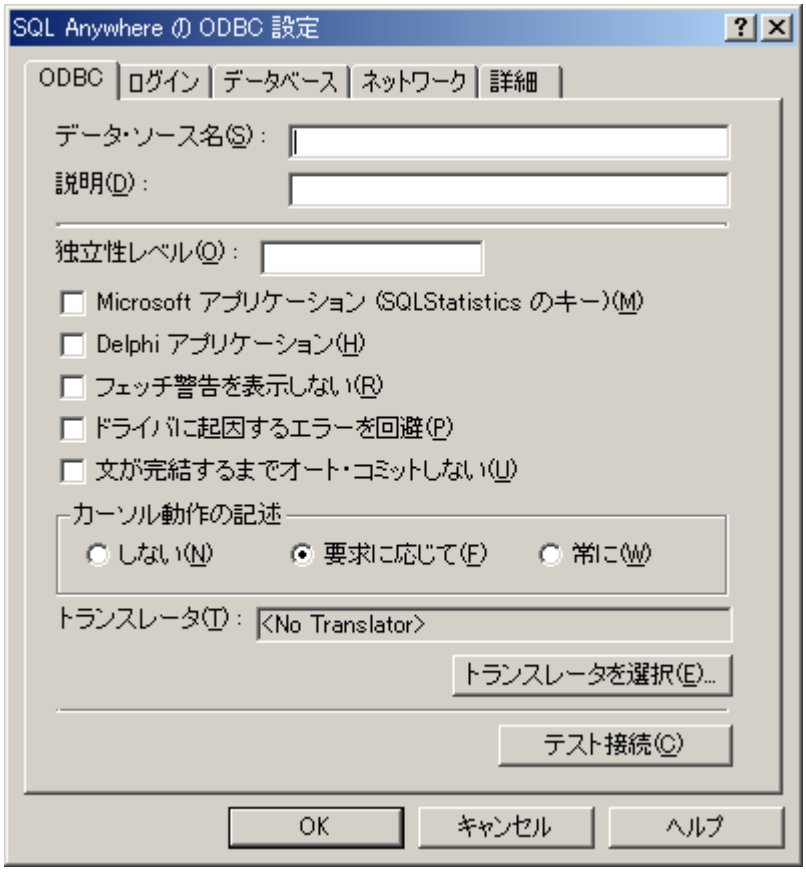
   [ODBC データ ソース アドミニストレータ] ダイアログが表示されます。

2. [追加] をクリックします。

   [データ ソースの新規作成] ウィザードが表示されます。

3. ドライバのリストから [SQL Anywhere 10] を選択し、[完了] をクリックします。

   [SQL Anywhere 10 の ODBC 設定] ダイアログが表示されます。



このダイアログに表示されるフィールドの詳細については、「[ODBC 設定] ダイアログのヘルプ」『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

4. 必要なパラメータを指定したら、[OK] をクリックしてダイアログを閉じ、データ・ソースを作成します。

ODBC アドミニストレータを使用した ODBC データ・ソースの設定

データ・ソースを編集するには、[ODBC アドミニストレータ] メイン・ウィンドウから必要なデータ・ソースを探して選択し、[設定] をクリックします。

コマンド・ラインからの ODBC データ・ソースの作成

dbdsn ユーティリティを使用して、ユーザ・データ・ソースとシステム・データ・ソースを作成できます。この方法では、ファイル・データ・ソースを作成することはできません。システム・データ・ソースに対する権限は Windows オペレーティング・システムだけに制限されており、ファイル・データ・ソースの作成には ODBC アドミニストレータを使用します。

♦ **ODBC データ・ソースを作成するには、次の手順に従います (コマンド・ラインの場合)。**

1. コマンド・プロンプトを開きます。
2. dbdsn コマンドを入力して、使用する接続パラメータを指定します。

   たとえば、次のコマンドはサンプル・データベースのデータ・ソースを作成します。コマンドは、1 行に入力する必要があります。

   ```
   dbdsn -w "My DSN" "UID=DBA;PWD=sql;DBF=samples-dir¥demo.db"
   ```

   samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

dbdsn ユーティリティの詳細については、「データ・ソース・ユーティリティ (dbdsn)」 646 ページを参照してください。

Mac OS X での ODBC データ・ソースの作成

Mac OS X 上で ODBC データ・ソースを作成する前に、SQL Anywhere ODBC ドライバを追加する必要があります。

♦ **SQL Anywhere ODBC ドライバを追加するには、次の手順に従います。**

1. */Applications/Utilities* から ODBC アドミニストレータを起動します。
2. [ドライバ] タブを選択します。
3. [追加] をクリックします。
4. [説明] フィールドに **SQL Anywhere 10** と入力します。
5. [Choose] をクリックし、[Driver File Name] と [Setup File Name] の両方のフィールドで SQL Anywhere ODBC ドライバを選択します。デフォルトで、このファイルは */Applications/SQLAnywhere10/System/lib/dbodbc10_r.bundle* にあります。

   バンドル名の *_r* は、ドライバのスレッド・バージョンであることを示します。非スレッド・アプリケーションで使用するための非スレッド・バージョン (*dbodbc10.bundle*) もあります。
6. [OK] をクリックします。

SQL Anywhere ODBC ドライバを追加すると、データベースへの接続に使用する ODBC データ・ソースを作成できるようになります。

ODBC データ・ソースの作成

♦ **ODBC データ・ソースを作成するには、次の手順に従います。**

1. */Applications/Utilities* から ODBC アドミニストレータを起動します。

2. ODBC アドミニストレータで、[ユーザー DSN] タブをクリックし、[追加] をクリックします。
3. 使用できるドライバのリストから SQL Anywhere 10 ドライバを選択します。[OK] をクリックします。
4. [データ・ソース名] フィールドに **SQL Anywhere 10 Demo** と入力し、次のように接続パラメータを追加します。接続パラメータと値の大文字と小文字は区別されません。

キーワード	値
UID	DBA
PWD	sql
START	dbeng10
DBF	/Applications/SQLAnywhere10/System/demo.db
ThreadManager	ON

   接続パラメータの詳細については、「接続パラメータとネットワーク・プロトコル・オプション」 229 ページを参照してください。
5. [OK] をクリックします。
6. [適用] をクリックします。
7. [Command] キーを押しながら [Q] キーを押し、ODBC アドミニストレータを終了します。

または、テキスト・エディタを使用して情報を追加することもできます。ODBC 設定ファイルは、ホーム・ディレクトリの */Library/ODBC* にあります。ドライブ情報として *odbcinst.ini* ファイル、データ・ソース情報として *odbc.ini* ファイルがあります。

データ・ソース・ユーティリティ (dbdsn) を使用して、Mac OS X に ODBC データ・ソースを作成することも可能です。「データ・ソース・ユーティリティ (dbdsn)」 646 ページを参照してください。

Windows でのファイル・データ・ソースの使用

Windows オペレーティング・システムでは、ODBC データ・ソースを通常システム・レジストリに格納します。ファイル・データ・ソースは、ファイルとして保存されるデータ・ソースです。通常、Windows のファイル・データ・ソースの拡張子は *.dsn* です。ファイル・データ・ソースは複数のセクションから構成されていて、各セクションは角カッコで囲まれた名前で始まります。

ファイル・データ・ソースを使用して接続するには、FileDataSourceName (FILEDSN) 接続パラメータを使います。1 つの接続で DataSourceName (DSN) と FileDataSourceName (FILEDSN) の両方は使用できません。

配布可能なファイル・データ・ソース

ファイル・データ・ソースの利点の 1 つは、ファイルをユーザに配布できることです。ファイルがファイル・データ・ソースのデフォルト ロケーションにある場合、ODBC によって自動的に選択されます。このように、多くのユーザの接続管理が簡単になります。

♦ ODBC ファイル・データ・ソースを作成するには、次の手順に従います (ODBC アドミニストレータの場合)。

1. ODBC アドミニストレータを起動し、[ファイル DSN] タブをクリックし、[追加] をクリックします。
2. ドライバのリストから [SQL Anywhere 10] を選択し、[次へ] をクリックします。
3. データ・ソースの作成手順に従います。

Windows CE での ODBC データ・ソースの使用

Windows CE では、ODBC ドライバ・マネージャまたは ODBC アドミニストレータを提供していません。このプラットフォームでは、SQL Anywhere はファイルに保管されている ODBC データ・ソースを使用します。DSN または FILEDSN キーワードのどちらかを指定することによって、これらのデータ・ソース定義を使用できます。Windows CE の場合のみ、DSN と FILEDSN は同義語です。

データ・ソースのロケーション

Windows CE は、データ・ソース・ファイルをデバイスのルート・ディレクトリ *¥filename.dsn* で検索します。

各データ・ソース自体は 1 つのファイルに保管されます。このファイルはデータ・ソースと同じ名前で、拡張子 *.dsn* が付きます。

参照

- 「Windows でのファイル・データ・ソースの使用」 79 ページ

Windows CE データ・ソースの例

次は、Windows CE 用の ODBC データ・ソースの例です。

```
[ODBC]
DRIVER=¥windows¥dbodbc10.dll
UID=DBA
PWD=sql
Integrated=No
AutoStop=Yes
EngineName=SalesDB_remote
LINKS=tcpip(host=192.168.0.55;port=2638;dobroadcast=none)
LOG=¥sa_connection.txt
START=dbsrv10 -c 8M
```

参照

- 「Windows CE デバイスに接続するための ODBC データ・ソースの作成」 1041 ページ

UNIX での ODBC データ・ソースの使用

UNIX オペレーティング・システムでは、ODBC データ・ソースはシステム情報ファイルに保管されます。デフォルトでは、このファイル名は *.odbc.ini* ですが、任意の名前を付けられます。システム情報ファイルの場所は次の順序で検索されます。

1. ODBCINI 環境変数
2. ODBC_INI 環境変数
3. ODBCHOME 環境変数
4. HOME 環境変数
5. ユーザのホーム・ディレクトリ (~)
6. PATH 環境変数

注意
ODBCINI と ODBC_INI 環境変数は、システム情報ファイル (*.odbc.ini* またはそれ以外) を指しますが、ODBCHOME と HOME 環境変数は *.odbc.ini* ファイルのパスを指します。
ODBCINI と ODBC_INI はどちらも、ファイル名を含むフル・パスで指定します。システム情報ファイルが ODBCINI または ODBC_INI で指定されたディレクトリにある場合、名前が *.odbc.ini* である必要はありません。

次にシステム情報ファイルの例を示します。

```
[My Data Source]
ENG=myserver
CommLinks=tcpip(Host=hostname)
UID=DBA
PWD=sql
```

システム情報ファイルには、任意の接続パラメータを入力できます。「接続パラメータ」 230 ページを参照してください。

ネットワーク・プロトコル・オプションは、CommLinks (LINKS) パラメータの一部として追加されます。「ネットワーク・プロトコル・オプション」 265 ページを参照してください。

ODBC データ・ソースを UNIX 上で作成したり管理したりするには、dbdsn ユーティリティを使用します。

警告
SQL Anywhere データ・ソースだけを使用している場合を除き、UNIX 上でファイル非表示ユーティリティ (dbfhide) を使用して、システム情報ファイル (デフォルトのファイル名は *.odbc.ini*) に単純暗号化を追加しないでください。他のデータ・ソース (Mobile Link 同期化など) を使用する予定の場合、システム情報ファイルの内容を難読化すると、他のドライバが正しく機能しなくなることがあります。

参照

- 「ODBC データ・ソースの作成」 76 ページ
- 「データ・ソース・ユーティリティ (dbdsn)」 646 ページ

デスクトップ・アプリケーションからの Windows CE データベースへの接続

Sybase Central や Interactive SQL など、デスクトップ PC で実行中のアプリケーションから、Windows CE デバイスで実行中のデータベース・サーバに接続できます。デスクトップ・コンピュータと Windows CE デバイスの間の ActiveSync リンクを介した接続では、TCP/IP を使用します。

参照

- 「Windows CE デバイス上でのサーバの起動」 1039 ページ
- 「Windows CE デバイス用プロキシ・ポートの作成」 1040 ページ
- 「Windows CE デバイスに接続するための ODBC データ・ソースの作成」 1041 ページ
- 「Windows CE デバイスの IP アドレスの特定」 1043 ページ

OLE DB を使用したデータベースへの接続

OLE DB は、コンポーネント・オブジェクト・モデル (COM) を使用してさまざまなソースからのデータをアプリケーションで使用できるようにします。リレーショナル・データベースは、OLE DB を介してアクセスできるデータ・ソースのクラスに入っています。

この項では、OLE DB を使用して次の環境から SQL Anywhere データベースに接続する方法について説明します。

- Microsoft ActiveX データ・オブジェクト (ADO) は、OLE DB データ・ソース用のプログラミング・インタフェースを提供しています。Microsoft Visual Basic のようなプログラミング・ツールから SQL Anywhere にアクセスできます。
- Sybase PowerBuilder は、OLE DB データ・ソースにアクセスできます。また、SQL Anywhere を PowerBuilder OLE DB データベース・プロファイルとして使用できます。

この項では、Sybase PowerBuilder と Visual Basic のような Microsoft ADO 環境から OLE DB を使用する方法の概要について説明します。これは、ADO または OLE DB を使用するプログラミング方法の完全なマニュアルではありません。開発に関する情報の主要な情報源については、使用している開発ツールのマニュアルを参照してください。

参照

- 「OLE DB の概要」『SQL Anywhere サーバ - プログラミング』

OLE DB プロバイダ

アクセスする各種のデータ・ソースごとに OLE DB プロバイダが必要です。各「**OLE DB プロバイダ**」は、動的リンク・ライブラリです。SQL Anywhere へのアクセスに使用できる OLE DB プロバイダは 2 つあります。

- **Sybase SQL Anywhere OLE DB プロバイダ**　SQL Anywhere OLE DB プロバイダは、OLE DB データ・ソースとしての SQL Anywhere へのアクセスを提供します。ODBC コンポーネントは必要ではありません。このプロバイダの省略名は、**SAOLEDB** です。

  SAOLEDB プロバイダは、インストールされると自動的に登録を行います。この登録プロセスには、レジストリの COM セクションへのレジストリ・エントリの作成も含まれます。これによって、ADO は **SAOLEDB** プロバイダが呼び出されたときに DLL を見つけることができます。DLL のロケーションを変更した場合は、それを再度登録する必要があります。

- **ODBC 用の Microsoft OLE DB プロバイダ**　Microsoft から、省略名 **MSDASQL** という OLE DB プロバイダが提供されています。

  MSDASQL プロバイダは、ODBC データ・ソースを OLE DB データ・ソースとして表示させます。これには SQL Anywhere ODBC ドライバが必要です。

参照

- 「OLE DB の概要」『SQL Anywhere サーバ - プログラミング』

ADO からの接続

ADO は、オブジェクト指向型プログラミング・インタフェースです。ADO では、**Connection** オブジェクトがデータ・ソースとのユニークなセッションを表します。

次の Connection オブジェクト機能を使用して、接続を開始できます。

- プロバイダ名が格納されている Provider プロパティ。プロバイダ名を指定しない場合は、ADO は MSDASQL プロバイダを使用します。
- 接続文字列が格納されている ConnectionString プロパティ。このプロパティに格納されている SQL Anywhere 接続文字列は、ODBC ドライバと同じ方法で使用されます。ODBC データ・ソース名、または明示的な UserID、Password、DatabaseName、その他のパラメータを、他の接続文字列にあるように指定できます。
- 接続を開始する Open メソッド。

例

次の Visual Basic コードは、SQL Anywhere への OLE DB 接続を開始します。

```
' Declare the connection object
Dim myConn as New ADODB.Connection
myConn.Provider = "SAOLEDB"
myConn.ConnectionString = "DSN=SQL Anywhere 10 Demo"
myConn.Open
```

参照

- 「SQL Anywhere を使用した ADO プログラミング」『SQL Anywhere サーバ - プログラミング』

接続パラメータのヒント

接続パラメータでは、所定の作業を実行するための方法が複数示されることがよくあります。データベース・サーバが接続文字列によって開始される組み込みデータベースでは、特によくあります。たとえば、接続文字列によってデータベースを開始する場合、DatabaseName (DBN) 接続パラメータまたは DatabaseSwitches (DBS) パラメータを使用してデータベース名を指定できます。Database Name (DBN) 接続パラメータの使用をおすすめします。

次は、接続パラメータが競合した場合の対処方法をいくつか示します。

- **DBF を使用してデータベース・ファイルを指定する**　StartLine (START) パラメータにデータベース・ファイルを指定するか、DatabaseFile (DBF) 接続パラメータを使用してデータベース・ファイルを指定できます。推奨は、DatabaseFile (DBF) パラメータです。
- **DBN を使用してデータベース名を指定する**　StartLine (START) パラメータか DatabaseSwitches (DBS) パラメータにデータベース名を指定するか、DatabaseName (DBN) 接続パラメータを使用してデータベース名を指定できます。推奨は、DatabaseName (DBN) パラメータです。
- **ENG を使用してデータベース・サーバ名を指定する**　まだ実行していないデータベース・ファイルを自動的に起動する場合、EngineName (ENG) パラメータにデータベース・サーバの名前を指定できます。これにより、データベースは、指定したデータベース・サーバに接続されます。
- **Start パラメータを使用してキャッシュ・サイズを指定する**　DatabaseFile (DBF) 接続パラメータを使用してデータベース・ファイルを指定しても、開始方法のチューニングが必要になる場合があります。これには StartLine (START) 接続パラメータを使用できます。

  たとえば、StartLine (START) 接続パラメータに追加のキャッシュ・メモリを指定する場合を考えてみます。次に示す組み込みデータベース接続パラメータの例は、追加のキャッシュを使用してデータベース・サーバを起動します。

```
DBF=samples-dir¥demo.db
DBN=Sample
ENG=Sample Server
UID=DBA
PWD=sql
START=dbeng10 -c 8M
```

接続パラメータについての注意

- **ブール値**　ブール (true または false) 引数は、true の場合は YES、ON、1、TRUE、Y、T のいずれかであり、false の場合は NO、OFF、0、FALSE、N、F のいずれかです。
- **大文字と小文字の区別**　値に大文字と小文字が区別されるもの (UNIX のファイル名など) が含まれている場合でも、接続パラメータは大文字と小文字を区別しません。
- インタフェース・ライブラリで使用される接続パラメータは、次の場所から取得できます (優先度の高い順)。
  - **接続文字列**　パラメータを接続文字列に明示的に渡すことができます。
  - **SQLCONNECT 環境変数**　SQLCONNECT 環境変数に接続文字列を格納できます。

- **データ・ソース**　ODBC データ・ソースにパラメータを格納できます。

- **文字セット制限**　サーバ名は、1 ー 127 の範囲の ASCII 文字セットで構成することをおすすめします。他のパラメータには、このような制限はありません。

- **優先度**
  次の規則によって、パラメータの優先度が決まります。

  - 接続文字列のエントリは、左から右に読み込まれます。同じパラメータが複数回指定された場合は、文字列の最後のパラメータを適用します。ODBC、OLE DB、Sybase Central、Interactive SQL、SQL Anywhere コンソール・ユーティリティはこの例外です。この場合、同じパラメータが複数回指定された場合は、最初の文字列を適用します。

  - 文字列にデータ・ソースやファイル・データ・ソースのエントリが含まれている場合、プロファイルは設定ファイルから読み込まれ、まだ設定されていない場合は、ファイルのエントリが使用されます。たとえば、接続文字列にデータ・ソース名があり、データ・ソースにあるいくつかのパラメータが明示的に設定されているときに競合が発生した場合、明示的なパラメータが使用されます。

- **接続文字列の解析**　接続文字列の解析中に問題が発生した場合、問題の発生源が接続パラメータであることを示すエラーが生成されます。

- **空の接続パラメータ**
  指定されている値が空の接続パラメータは、長さ 0 の文字列として扱われます。

参照

- 「接続パラメータとネットワーク・プロトコル・オプション」 229 ページ
- 「接続文字列と文字セット」 344 ページ

接続のトラブルシューティング

この項の対象読者

多くの場合、この章の最初の部分で説明されている情報を活用して、データベースに簡単に接続できます。

ただし、サーバへの接続の確立に問題がある場合は、それを解決するために、SQL Anywhere が接続を確立するプロセスについて理解する必要があります。この項では、SQL Anywhere の接続動作について説明します。

ファイアウォールを介した接続を含むネットワーク固有の問題については、「クライアント／サーバ通信」 121 ページを参照してください。

ソフトウェアは、次に示す各クライアント・アプリケーションの場合とまったく同じ手順に従います。

- **ODBC** SQLDriverConnect 関数を使用する ODBC アプリケーションです。これは ODBC アプリケーションでの一般的な接続方法です。Sybase PowerBuilder などのアプリケーション開発システムの多くは、このクラスのアプリケーションに属します。SQLConnect 関数は、ODBC アプリケーションでも使用できます。
- **Embedded SQL** Embedded SQL を使用し、データベースとの接続で推奨関数 (db_string_connect) を使用するクライアント・アプリケーションです。

  さらに、Embedded SQL アプリケーションと Interactive SQL では、SQL CONNECT 文も使用できます。これには、CONNECT AS… と CONNECT USING… の 2 つの書式があります。Interactive SQL を含むすべてのデータベース管理ツールは、db_string_connect を使用します。
- **OLE DB** ADODB Connection オブジェクトを使用する任意の ADO アプリケーション。Provider プロパティは、OLE DB ドライバの場所を指定するために使用されます。Connection String プロパティでは、**DataSourceName** の代わりに **DataSource** が、**UserID** の代わりに **User ID** が使用されることがあります。
- **JDBC** iAnywhere JDBC ドライバを使用するアプリケーションは、Driver Manager.GetConnection メソッドのパラメータとして URL **jdbc:ianywhere:** の後に標準の接続文字列を渡します。接続文字列には、**DataSource=** を含めて SQL Anywhere データ・ソースを指定するか、**Driver=SQL Anywhere 10** (UNIX と Linux では、このパラメータは **Driver=libdbodbc10** のように指定) を含めてください。

参照

- 「サーバ起動時のトラブルシューティング」 54 ページ
- 「ネットワーク通信のトラブルシューティング」 132 ページ

接続を確立する手順

接続を確立するため、SQL Anywhere は次の手順を実行します。

1. **インタフェース・ライブラリの検出**　クライアント・アプリケーションは、インタフェース・ライブラリを見つけます。
2. **接続パラメータ・リストのアセンブル**　接続パラメータはさまざまな場所 (データ・ソース、アプリケーションによってアセンブルされる接続文字列、環境変数など) に設定されるので、SQL Anywhere は、複数のパラメータを 1 つのリストにアセンブルします。
3. **サーバの検出**　SQL Anywhere は、接続パラメータを使用して、コンピュータまたはネットワーク上のデータベース・サーバを検出します。
4. **データベースの検出**　SQL Anywhere は、接続先となるデータベースを検出します。
5. **パーソナル・サーバの起動**　SQL Anywhere は、サーバの検出に失敗した場合は、パーソナル・データベース・サーバを起動してデータベースをロードしようとします。

インタフェース・ライブラリの検出

クライアント・アプリケーションは、SQL Anywhere インタフェース・ライブラリの 1 つを呼び出します。通常、この DLL または共有ライブラリのロケーションは、ユーザには見えません。この項では、問題が発生した場合のライブラリの検出方法について説明します。

ODBC ドライバのロケーション

ODBC では、インタフェース・ライブラリは ODBC ドライバとも呼ばれます。ODBC クライアント・アプリケーションが ODBC ドライバ・マネージャを呼び出し、ドライバ・マネージャが SQL Anywhere ドライバを見つけます。

ODBC ドライバ・マネージャは、ドライバを見つけるため、指定されたデータ・ソースの中を見ます。ODBC アドミニストレータまたは dbdsn ユーティリティを使用してデータ・ソースを作成すると、SQL Anywhere は ODBC ドライバの現在のロケーションを書き込みます。データ・ソース情報は、Windows ではレジストリに格納されます。UNIX ではファイル (デフォルトでは *.odbc.ini* という名前のファイル) に格納されます。

Embedded SQL インタフェース・ライブラリのロケーション

Embedded SQL アプリケーションは、インタフェース・ライブラリを名前で指定して呼び出します。SQL Anywhere の Embedded SQL インタフェース・ライブラリの名前は次のとおりです。

- **Windows**　*dblib10.dll*
- **UNIX**　オペレーティング・システム固有の拡張子を持つ *dblib10*
- **NetWare**　*dblib10.nlm*

OLE DB ドライバのロケーション

SQL Anywhere OLE DB プロバイダ DLL (*dboledb10.dll)* の検索には、レジストリのエントリに基づき、プロバイダ名 (SAOLEDB) が使用されます。エントリは、SAOLEDB のインストール時、または再登録時に作成されます。

ADO.NET

ADO.NET プログラムは SQL Anywhere ADO.NET プロバイダ (*iAnywhere.Data.SQLAnywhere.dll*) への参照を追加します。.NET データ・プロバイダ DLL は、インストール時に .NET グローバル・アセンブリ・キャッシュ (GAC) に追加されます。

iAnywhere JDBC ドライバのロケーション

アプリケーションを実行するとき、Java パッケージ *jodbc.jar* がクラス・パスにあることが必要です。ネイティブ DLL または共有オブジェクトがシステムによって検出される必要があります。

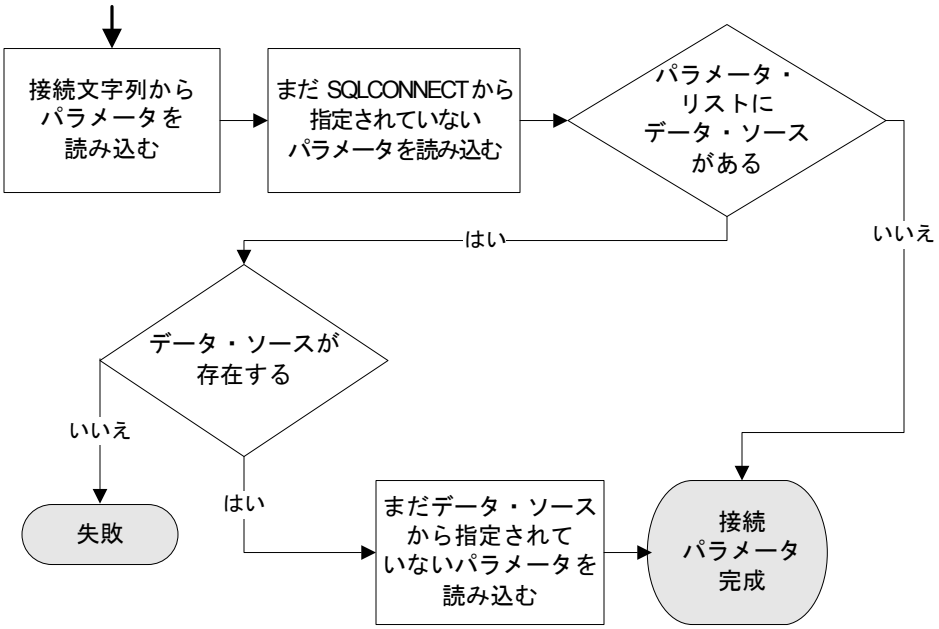
- **PC オペレーティング・システム** Windows などの PC オペレーティング・システムでは、現在のディレクトリ、システム・パス、*Windows* ディレクトリ、*Windows¥system* ディレクトリからファイルを探します。
- **UNIX オペレーティング・システム** UNIX では、システム・パスとユーザ・ライブラリ・パスからファイルを探します。
- **NetWare** NetWare では、検索パスと *sys:system* ディレクトリからファイルを探します。

ライブラリが検出されるタイミング

クライアント・アプリケーションは、インタフェース・ライブラリを検出すると、それに接続文字列を渡します。インタフェース・ライブラリは接続文字列を使用して接続パラメータのリストをアセンブルし、これをサーバとの接続を確立するときに使用します。

接続パラメータ・リストのアセンブル

次の図は、インタフェース・ライブラリが接続の確立に使用する接続パラメータのリストをアセンブルする方法を示しています。



注意

図の重要な点を次に示します。

- **優先度** 複数の場所に格納されているパラメータは次の優先順位に従います。

  1. 接続文字列
  2. SQLCONNECT
  3. データ・ソース

  つまり、パラメータがデータ・ソースと接続文字列の両方に指定された場合、接続文字列の値がデータ・ソースの値よりも優先されます。

- **失敗** この段階で失敗するのは、存在しないデータ・ソースが、接続文字列または SQLCONNECT の中に指定されている場合だけです。

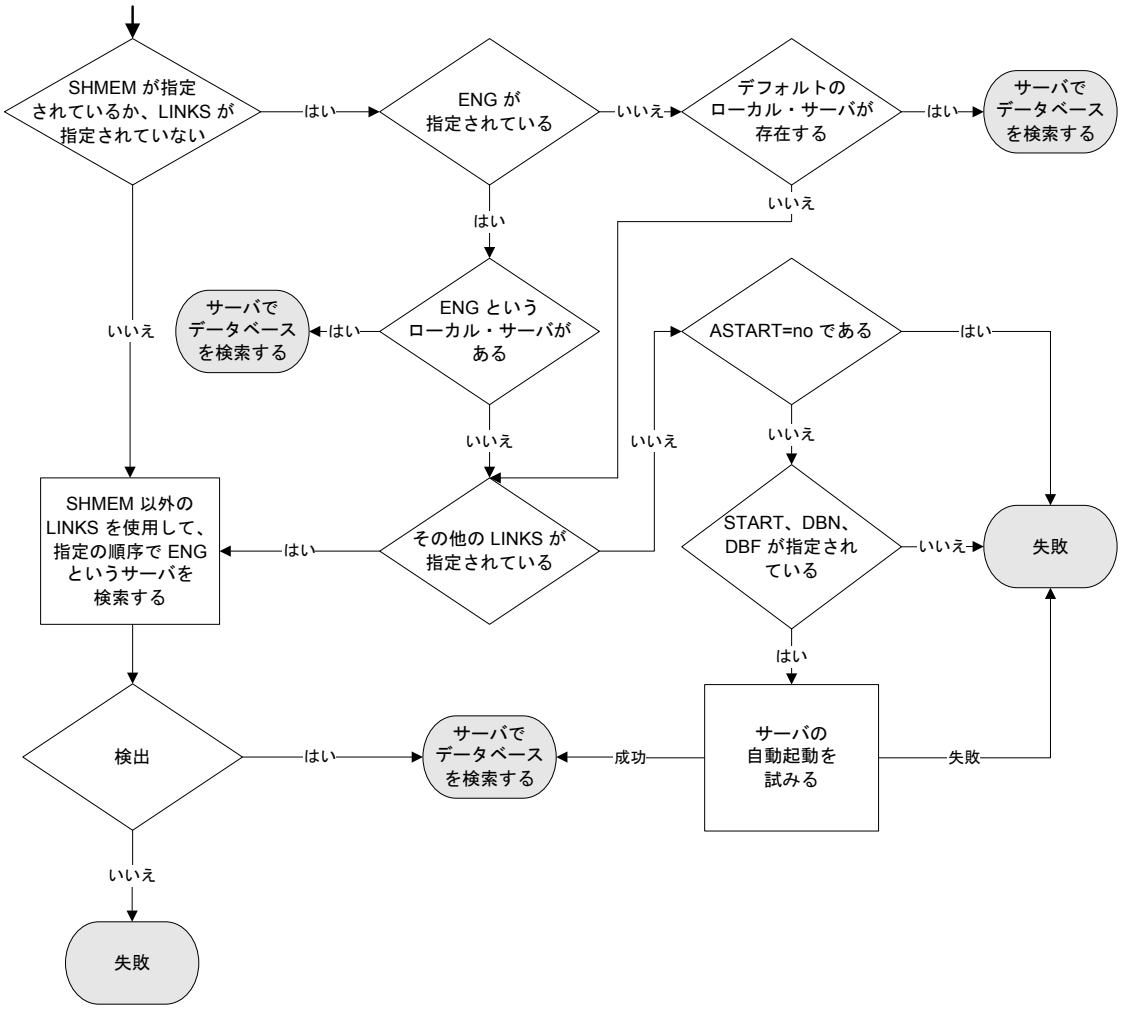
- **共通パラメータ** すでに使用されている他の接続によっては、一部の接続パラメータを無視するものがあります。これには、次のパラメータが含まれます。

  - **Autostop** データベースがすでにロードされている場合は無視されます。
  - **DatabaseFile** DatabaseName が指定され、この名前を持つデータベースがすでに実行されている場合は無視されます。

インタフェース・ライブラリは、アセンブル後の接続パラメータのリストを使用して接続を試行します。

データベース・サーバの検出

接続確立処理の次の段階では、SQL Anywhere はサーバを検出しようとします。接続パラメータ・リストにサーバ名 (EngineName (ENG) 接続パラメータ) が含まれる場合、その名前のサーバの検索が実行されます。EngineName (ENG) 接続パラメータが指定されておらず、LINKS が指定されていないか LINKS に共有メモリが含まれる場合、SQL Anywhere はデフォルト・サーバを探します。



SQL Anywhere はサーバを検出すると、必要なデータベースをそのサーバ上で検出またはロードしようとします。「データベースの検出」 94 ページを参照してください。

SQL Anywhere は、サーバを検出できないと、接続パラメータに応じてパーソナル・サーバを起動しようとする場合があります。

注意

- ローカル接続では、サーバの検出は簡単です。ネットワークを介した接続では、CommLinks (LINKS) 接続パラメータを使用して、ネットワーク・プロトコル・オプションを指定することで、検索方法をチューニングできます。
- CommLinks (LINKS) 接続パラメータへの引数には、各ネットワーク・プロトコルに対してネットワーク・プロトコル・オプションのセットを指定できます。
- サーバの検索には 2 つの手順があります。SQL Anywhere は、まず、その名前のサーバが使用できるかどうかを確認するためにサーバ名キャッシュを調べます (DoBroadcast の値が none の場合はこの手順を省略します)。次に、使用可能な接続パラメータを使用して接続を試行します。
- サーバが自動的に起動される場合は、DBF、DBKEY、DBS、DBN、ENG、および AUTOSTOP 接続パラメータの情報を使用して、そのサーバのオプションが構築されます。
- サーバが代替サーバ名を持っている場合、代替サーバ名を指定して起動したデータベースへの接続にのみ代替サーバ名を使用できます。そのデータベース・サーバ上で実行中の他のデータベースに接続するのに代替サーバ名を使用することはできません。「-sn オプション」 225 ページを参照してください。

SQL Anywhere Broadcast Repeater を使用したデータベース・サーバの検出

SQL Anywhere Broadcast Repeater を使用すると、SQL Anywhere クライアントにおいて、HOST 接続パラメータや LDAP を使用しなくても、他のサブネット上で通常は UDP ブロードキャストが到達できないファイアウォールを介して動作している SQL Anywhere データベース・サーバを検出することができます。

♦ SQL Anywhere Broadcast Repeater を使用するには、次の手順に従います。

1. サブネット内の任意のコンピュータで DBNS (データベース・ネーム・サービス) プロセスを起動します。
2. 別のサブネット内の任意のコンピュータで DBNS プロセスを起動し、最初のコンピュータのコンピュータ名または IP アドレスをパラメータとして渡します (*address* パラメータを使用します)。

   2 つの DBNS プロセスが TCP/IP 相互接続を確立します。
3. いずれの DBNS プロセスもそれぞれのサブネット上でブロードキャストを受信します。各 DBNS プロセスが TCP/IP 接続上で他方の DBNS プロセスに要求を送信し、受信した DBNS プロセスはそのサブネット上で要求を再ブロードキャストし、さらに送信元の DBNS プロセスに応答を送信します。送信元の DBNS プロセスは、受信した応答を要求元のクライアントに送信します。

4. 通常の SQL Anywhere ブロードキャストは、リモート・サブネット上のサーバに到達し、クライアントは HOST パラメータを指定することなくリモート・サブネット上のサーバに接続できます。

相互通信が可能な DBNS プロセスの数に制限はありません。各 DBNS プロセスが検出したすべての DBNS プロセスに接続し、複数の DBNS プロセスが DBNS プロセス・リストを共有します。たとえば、A と B という 2 つの DBNS プロセスを起動したとします。これらとは別の第 3 のサブネットで DBNS プロセス C を起動し、プロセス B のアドレスをプロセス C に渡すと、プロセス B がプロセス C にプロセス A の存在を通知し、プロセス C がプロセス A に接続します。

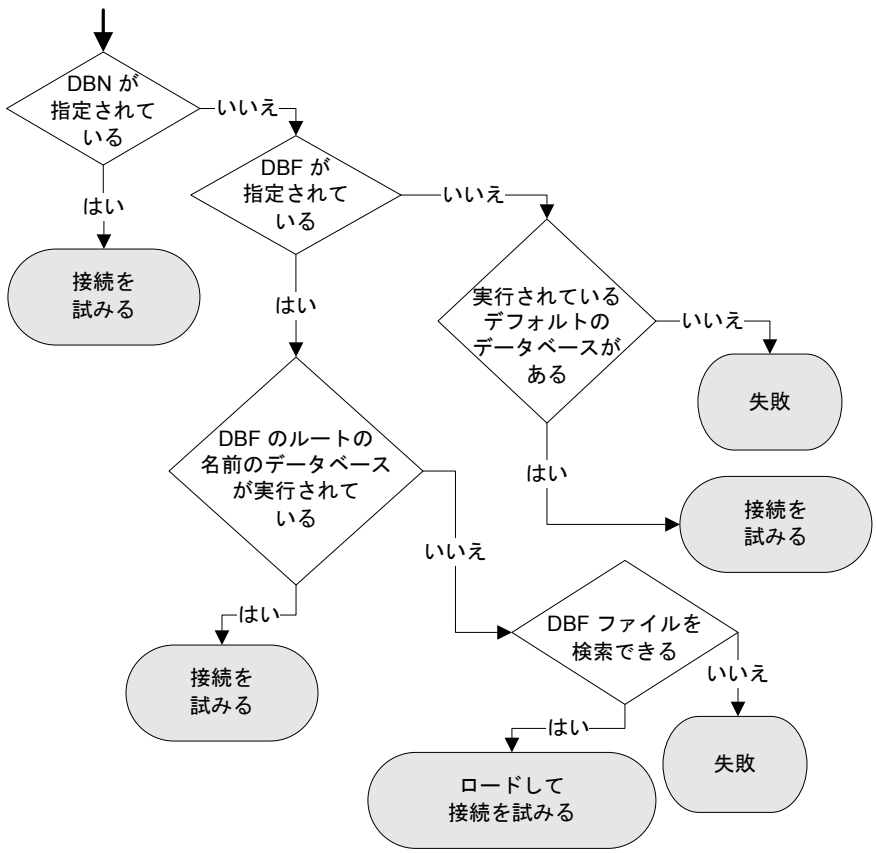
1 つのサブネットで複数の DBNS プロセスを実行することは不要であり、推奨できません。

参照

- 「SQL Anywhere Broadcast Repeater ユーティリティ (dbns10)」 715 ページ

データベースの検出

SQL Anywhere は、サーバの検出に成功すると、データベースの検出に移ります。次に例を示します。



迅速な接続のためのサーバ名キャッシュ

DoBroadcast (DOBROAD) プロトコル・オプションが DIRECT または ALL に設定されると、ネットワーク・ライブラリは CommLinks (LINKS) 接続パラメータを使ってネットワーク上をブロードキャストすることでデータベース・サーバを検索します。

ブロードキャストのチューニング

CommLinks (LINKS) パラメータは、使用するプロトコルをリストした文字列を引数としてとります。さらにオプションで、ブロードキャストをチューニングする各種ネットワーク・プロトコル・オプションも引数としてとることができます。「ネットワーク・プロトコル・オプション」 265 ページを参照してください。

サーバ情報のキャッシュ

大規模なネットワーク上をブロードキャストして特定の名前のサーバを検索するには、時間がかかります。サーバ・アドレスをキャッシュすると、サーバへの最初の接続が見つかったプロトコルとそのアドレスがファイルに保存され、後続の接続でその情報使用することで、ネットワーク接続が高速化されます。

サーバ情報は *sasrv.ini* というキャッシュ・ファイルに保存されます。このファイルには一連のセクションがあり、それぞれが次の形式になっています。

```
[Server name]
LINKS=protocol_name
Address=address_string
```

sasrv.ini のデフォルトのロケーションは、Windows では *%ALLUSERSPROFILE%¥Application Data¥SQL Anywhere 10*、UNIX では *~/.sqlanywhere10* です。

注意
それぞれのサーバの名前がユニークであることは、非常に重要です。異なるサーバに同じ名前を付けると、識別の問題を引き起こす可能性があります。

キャッシュの使用方法

キャッシュ内のサーバ名とプロトコルが接続文字列と一致する場合、SQL Anywhere は、まずキャッシュ・アドレスを使って接続を試みます。接続に失敗した場合、またはキャッシュ内のサーバ名とプロトコル名が接続文字列と一致しない場合、ブロードキャストを使ったサーバの検索には接続文字列情報が使用されます。ブロードキャストが成功すると、キャッシュ内のサーバ名エントリが上書きされます。サーバが見つからなかった場合、キャッシュ内のサーバ名エントリは削除されます。DoBroadcast プロトコル・オプションが none に設定されている場合、キャッシュされたアドレスはすべて無視されます。

Interactive SQL 接続

データベースにすでに接続しているときに CONNECT 文を発行した場合、Interactive SQL の動作は Embedded SQL のデフォルトの動作とは異なります。CONNECT 文にデータベースやサーバが指定されていなければ、Interactive SQL は、デフォルト・データベースではなく現在のデータ

ベースに接続します。この動作は、データベースを再ロードするときに必要です。「CONNECT 文 [ESQL] [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

サーバを見つけられるかどうかのテスト

dbping ユーティリティは、接続のトラブルシューティングで役に立ちます。特に、特定の名前のサーバがネットワーク上で使用可能かどうかテストするために使用できます。

dbping ユーティリティは、接続文字列をオプションとして指定できますが、デフォルトではサーバを見つけるために必要な部分だけを使用します。このユーティリティは、デフォルトではサーバを起動しませんが、-d オプションが指定されている場合は、起動を試行することがあります。

例

次のコマンド・ラインは、Waterloo というサーバが TCP/IP 接続で使用できるかどうかをテストします。

```
dbping -c "ENG=Waterloo;CommLinks=tcpip"
```

次のコマンドは、デフォルト・サーバが現在のコンピュータ上で使用できるかどうかをテストします。

```
dbping
```

参照

- 「Ping ユーティリティ (dbping)」 693 ページ

Embedded SQL 接続のパフォーマンスのテスト

Embedded SQL 接続のパフォーマンスに関する情報を取得するには、ping ユーティリティ (dbping) を使用し、-s または -st オプションを指定します。次の統計値が収集されます。

統計情報	説明
DBLib の接続と切断	DBLib の接続および切断を 1 回実行する時間。ODBC などの他のインタフェースを使用した接続および切断は、接続の完了に必要な要求が多く、パフォーマンスは DBLib より低くなるのが一般的です。
簡単な要求の往復時間	クライアントからサーバに要求を送信するのにかかる時間とサーバからクライアントに応答を送信するのにかかる時間との和。往復時間は平均遅延時間の 2 倍になります。
送信スループット	dbping からデータベース・サーバへの反復ごとに 100 KB のデータを転送する場合のスループット。
受信スループット	データベース・サーバから dbping への反復ごとに 100 KB のデータを転送する場合のスループット。

往復時間が長く、スループットが高いネットワークの場合、往復時間が長いために、レポートされるスループットはネットワークの実際のスループットより低くなります。通信圧縮によりパ

フォーマンスが向上するかどうかを確認するには、dbping -s を使用すると便利です。パフォーマンス統計は概算値であり、クライアント・コンピュータとサーバ・コンピュータの両方がアイドル状態である方が統計値の精度は高くなります。通信圧縮を使用すると、送信されるデータは元のサイズの約 25% に圧縮されます。

dbping -s コマンドからの出力例を次に示します。実行した dbping コマンドは dbping -s -c "UID=DBA;PWD=sql;ENG=sampleserver;LINKS=TCPIP" です。

```
SQL Anywhere Server Ping Utility Version 10.0.0.1658
Connected to SQL Anywhere 10.0.0.1657 server "sampleserver" and database "sample"
at address 10.25.107.108.
Performance statistic          Number          Total Time    Average
----------------------------   --------------   ----------   ------------
DBLib connect and disconnect      175 times       1024 msec        5 msec
Round trip simple request        2050 requests    1024 msec       <1 msec
Send throughput                  7600 KB          1024 msec     7421 KB/sec
Receive throughput               10100 KB         1024 msec     9863 KB/sec
Ping database successful.
```

参照

- 「Ping ユーティリティ (dbping)」 693 ページ

統合化ログインの使用方法

「**統合化ログイン**」機能を使用すると、データベース接続とオペレーティング・システムやネットワークのログインの両方を、単一のユーザ ID とパスワードで管理できます。この項では、統合化ログイン機能について説明します。

サポートされるオペレーティング・システム

統合化ログイン機能は、Windows で実行されているデータベース・サーバで使用できます。統合化ログインを使用すると、Windows クライアントが Windows で実行されているネットワーク・サーバに接続できます。

統合化ログインの利点

統合化ログインは、1 つまたは複数の Windows ユーザ・プロファイルまたは Windows ユーザ・グループ・プロファイルからデータベース内の既存のユーザへのマッピングです。ユーザ・プロファイルまたはグループのセキュリティをナビゲートし、コンピュータへのログインに成功したユーザは、他のユーザ ID またはパスワードを指定しないでデータベースに接続できます。

そのためには、統合化ログインを使用するようにデータベースを設定し、コンピュータやネットワークへのログインに使用するユーザまたはグループのプロファイルとデータベース・ユーザの間のマッピングを許可します。

統合化ログインの使用は、ユーザにとって便利であると同時に、1 つのセキュリティ・システムでデータベースとネットワークの両方のセキュリティを維持できます。次のような利点があります。

- ユーザによるユーザ ID とパスワードの入力は不要です。
- ユーザの認証はデータベースではなくオペレーティング・システムによって行われます。データベースのセキュリティと、コンピュータやネットワークのセキュリティには、単一のシステムが使用されます。
- 複数のユーザまたはグループ・プロファイルを 1 つのデータベース・ユーザ ID にマッピングできます。
- Windows コンピュータへのログインに使用する名前とパスワードは、データベース・ユーザの ID とパスワードと一致している必要はありません。

警告
統合化ログインにはセキュリティ・システムが単一であるという利便性がありますが、そのためには、データベース管理者がセキュリティ上の重要事項を熟知しておく必要があります。「セキュリティについての考慮事項：無制限データベース・アクセス」 106 ページと「セキュリティについての考慮事項：コピーされたデータベース・ファイル」 119 ページを参照してください。

Windows ユーザ・グループ用の統合化ログインの作成

各 Windows ユーザに対して統合化ログインを作成することに加え、Windows ユーザ・グループに対しても統合化ログインを作成できます。

Windows ユーザがログインする時、明示的な統合化ログイン・マッピングを持っていないが、統合化ログイン・マッピングがある Windows ユーザ・グループに所属している場合、そのユーザは、Windows ユーザ・グループの統合化ログイン・マッピングで指定されたデータベース・ユーザまたはグループとしてデータベースに接続します。

警告
Windows ユーザ・グループの統合化ログインを作成すると、そのグループに属するすべてのユーザが、ユーザ ID やパスワードを知らなくてもデータベースに接続できます。
「Windows ユーザ・グループのメンバによるデータベースへの接続を防ぐ」 100 ページを参照してください。

複数グループのメンバ

Windows ユーザが複数の Windows ユーザ・グループに属し、コンピュータ上の複数の Windows ユーザ・グループの統合化ログイン・マッピングがデータベースに存在する場合、統合化ログインが成功するのは、コンピュータ上の Windows ユーザ・グループのすべてが同じデータベース・ユーザ ID に対する統合化ログイン・マッピングを持っている場合のみです。複数の Windows ユーザ・グループが異なるデータベース・ユーザ ID への統合化ログイン・マッピングを持つ場合は、エラーが返され、統合化ログインは失敗します。

たとえば、dbuserA と dbuserB という 2 つのユーザ ID があるデータベースと、Windows ユーザ・グループ xpgroupA と xpgroupB に属する Windows ユーザ windowsuser について考えてみます。

SQL 文	許可内容
GRANT INTEGRATED LOGIN TO windowsuser AS USER dbuserA	windowsuser に対して明示的に設定された統合化ログイン・マッピングを使用して windowsuser はデータベースに接続します。
GRANT INTEGRATED LOGIN TO xpgroupA AS USER dbuserB	windowsuser は、xpgroupA に対して付与された統合化ログイン・マッピングを使用してデータベースに接続します。
GRANT INTEGRATED LOGIN TO xpgroupA AS USER dbuserB; GRANT INTEGRATED LOGIN xpgroupb AS USER dbuserB	windowsuser が属する両方の Windows ユーザ・グループに同じデータベース・ユーザへの統合化ログイン・マッピングがあるため、windowsuser はデータベースに接続できます。

SQL 文	許可内容
GRANT INTEGRATED LOGIN TO xpgroupA AS USER dbuserA; GRANT INTEGRATED LOGIN TO xpgroupb AS USER dbuserB	データベースへは接続できません。windowsuser がデータベースに接続しようとした場合、各 Windows ユーザ・グループには異なるデータベース・ユーザへの統合化ログイン・マッピングがあり、windowsuser は両方の Windows ユーザ・グループのメンバであるため、統合化ログインは失敗します。

Domain Controller のロケーション

デフォルトでは、Windows ユーザ・グループのメンバシップを確認するために、SQL Anywhere データベース・サーバを実行中のコンピュータが使用されます。Domain Controller サーバが、データベース・サーバを実行中のコンピュータとは異なる場合は、integrated_server_name オプションを使用して、Domain Controller サーバの名前を指定できます。次に例を示します。

```
SET PUBLIC.OPTION integrated_server_name = '¥¥myserver-1'
```

「integrated_server_name オプション [データベース]」 454 ページを参照してください。

Windows ユーザ・グループのメンバによるデータベースへの接続を防ぐ

Windows ユーザ・グループの統合化ログインを作成すると、そのグループに属するすべてのユーザが、データベース・ユーザ ID やパスワードを知らなくてもデータベースに接続できます。統合化ログインを持つ Windows ユーザ・グループのメンバであるユーザが、グループ統合化ログインを使用してデータベースに接続できないようにするには、2 つの方法があります。

- パスワードのないデータベース・ユーザ ID に対して、ユーザの統合化ログインを作成します。
- ユーザがログインを許可されているかどうかを確認し、無許可のユーザが接続しようとすると例外を発行するストアド・プロシージャを作成します。このストアド・プロシージャは、[login_procedure] オプションによって呼び出します。

これらのどちらの方法を使用しても、Windows ユーザ・グループのメンバがデータベースへ接続できないようにします。

パスワードなしのユーザ ID に統合化ログインを作成する

ユーザが統合化ログインを持つ Windows ユーザ・グループのメンバで、さらにそのユーザ ID に対する明示的な統合化ログインも持っている場合、データベースへの接続にはそのユーザの統合化ログインが使用されます。ユーザが Windows ユーザ・グループの統合化ログインを使用してデータベースへ接続できないようにするには、データベース・ユーザ ID への Windows ユーザの統合化ログインをパスワードなしで作成します。パスワードを持たないデータベース・ユーザ ID は、データベースに接続できません。

♦ **パスワードなしのユーザ ID に統合化ログインを作成するには**

1. データベースにパスワードなしでユーザを追加します。次に例を示します。

```
GRANT CONNECT TO db_user_no_password
```

2. パスワードを持たないデータベース・ユーザにマップする Windows ユーザの統合化ログインを作成します。次に例を示します。

```
GRANT INTEGRATED LOGIN TO WindowsUser
AS USER db_user_no_password
```

Windows ユーザの接続を防ぐ手順の作成

login_procedure オプションは、データベースへの接続が試行されるたびに呼び出されるストアド・プロシージャを指定します。デフォルトでは dbo.sp_login_environment プロシージャが呼び出されます。login_procedure オプションを選択すると、特定のユーザがデータベースに接続できないようにするために作成したプロシージャを呼び出すことができます。

次の例では、login_procedure オプションによって呼び出される login_check というプロシージャを作成します。login_check プロシージャは、データベースに接続できないユーザのリストに照らし合わせて、指定されたユーザ名を確認します。指定されたユーザ名がリストに見つかると、接続は失敗します。この例では、Joe、Harry、または Martha というユーザは接続を許可されていません。ユーザがリストに見つからない場合、データベース接続は通常どおり実行され、sp_login_environment プロシージャが呼び出されます。

```
CREATE PROCEDURE DBA.user_login_check()
  BEGIN
    DECLARE INVALID_LOGON EXCEPTION FOR SQLSTATE '28000';
    // Disallow certain users
    IF( CURRENT USER IN ('Joe','Harry','Martha') ) THEN
      SIGNAL INVALID_LOGON;
    ELSE
      CALL sp_login_environment;
    END IF;
  END
go
GRANT EXECUTE ON DBA.user_login_check TO PUBLIC
go
SET OPTION PUBLIC.login_procedure='DBA.user_login_check'
go
```

統合化ログインの設定

統合化ログインを使用して正常に接続するには、いくつかの手順を実行してください。

♦ **統合化ログインを使用するには、次の手順に従います。**

1. login_mode データベース・オプションの値を、デフォルト値である Standard の代わりに Integrated (大文字と小文字の区別はなし) を含む設定にすることによって、データベース内で統合化ログイン機能が有効になります。この手順には、DBA 権限が必要です。

2. 統合化ログインのマッピング先となるデータベース・ユーザを作成します (存在しない場合)。これは、SQL 文を使用するか、Sybase Central でウィザードを使用して実行できます。この手順には、DBA 権限が必要です。
3. Windows ユーザまたはグループ・プロファイルと既存のデータベース・ユーザの間に、統合化ログイン・マッピングを作成します。これは、SQL 文を使用するか、Sybase Central でウィザードを使用して実行できます。Sybase Central では、統合化ログイン・パーミッションを持つユーザはすべて [ログイン・マッピング] フォルダに表示されます。この手順には、DBA 権限が必要です。
4. 統合化ログイン機能がトリガされる方法で、クライアント・アプリケーションから接続します。

以上の各手順については、この後の項で説明します。

統合化ログイン機能の有効化

login_mode データベース・オプションは、統合化ログイン機能が有効かどうかを判断します。データベース・オプションは、それが指定されているデータベースにしか適用されないため、同じサーバ内にロードされて動作している場合でも、データベースが異なれば統合化ログインの設定も異なります。

login_mode データベース・オプションに指定できる値は次のとおりです。

- **Standard**　標準ログインを許可します。これはデフォルト設定です。標準ログインでは、ユーザ ID とパスワードの両方を入力する必要があり、統合化接続や Kerberos 接続のパラメータは使用されません。統合化ログインまたは Kerberos ログインを使用して接続しようとすると、エラーが発生します。
- **Integrated**　統合化ログインを許可します。
- **Kerberos**　Kerberos ログインを許可します。「Kerberos 認証の使用」 108 ページを参照してください。

警告
login_mode データベース・オプションを標準ログインが許可されない設定にした場合、接続できるのは、統合化ログイン・マッピングまたは Kerberos ログイン・マッピングを付与されているユーザまたはグループだけに制限されます。ユーザ ID とパスワードで接続しようとすると、エラーが発生します。唯一の例外は、DBA 権限を持つユーザです。

複数のログインのタイプを許可するには、login_mode オプションに複数の値を指定します。たとえば、次の SQL 文は、標準ログインと統合化ログインの両方の接続を許可するよう login_mode データベース・オプションの値を設定します。

```
SET OPTION PUBLIC.login_mode = 'Standard,Integrated'
```

データベース・ファイルがコピー可能な場合、一時的なパブリック login_mode オプションを使用する必要があります (統合化ログインと Kerberos ログインの両方で)。ファイルをコピーする場合、統合化ログインと Kerberos ログインはデフォルトではサポートされません。

統合化ログインの作成

ユーザ・プロファイルは、既存のデータベース・ユーザ ID に対してのみマッピングできます。データベースからデータベース・ユーザ ID が削除されると、そのデータベース・ユーザ ID に基づくすべての統合化ログイン・マッピングも、自動的に削除されます。

1 つのデータベース・ユーザ ID に対して 1 つのユーザまたはグループ・プロファイルをマッピングする必要はありません。1 つ以上のユーザ・プロファイルを、同一のデータベース・ユーザ ID にマッピングできます。

DBA 権限を持つユーザのみ、統合化ログイン・マッピングを作成または削除できます。

統合化ログイン・マッピングは、Sybase Central でウィザードを使用するか、SQL 文を使用して作成できます。

♦ **統合化ログインをマッピングするには、次の手順に従います (Sybase Central の場合)。**

1. DBA ユーザとしてデータベースに接続します。
2. 左ウィンドウ枠にある [ログイン・マッピング] フォルダを開きます。
3. [ファイル] メニューから [新規] – [ログイン・マッピング] を選択します。
4. ウィザードの 2 番目のページに、統合化ログインが作成されるシステム (コンピュータ) のユーザ・プロファイル名またはグループ・プロファイル名を入力します。

   また、このユーザのマッピング先のデータベース・ユーザ ID を選択します。ウィザードに、使用可能なデータベース・ユーザが表示されます。その中の 1 つを選択してください。新しいデータベース・ユーザ ID は追加できません。
5. ウィザードの指示に従います。

♦ **統合化ログインをマッピングするには、次の手順に従います (SQL の場合)。**

1. DBA ユーザとしてデータベースに接続します。
2. GRANT INTEGRATED LOGIN TO 文を実行します。

例

次の SQL 文を使用すると、Window ユーザの fran_whitney と matthew_cobb は、DBA ユーザ ID またはパスワードを知らなかったり指定しなかったりしても、DBA ユーザとしてデータベースにログインできます。

```
GRANT INTEGRATED LOGIN
TO fran_whitney, matthew_cobb
AS USER DBA
```

「GRANT 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

次の SQL 文を使用すると、Windows NT グループの mywindowsusers のメンバである Windows ユーザは、DBA ユーザ ID たはパスワードを知らなかったり指定しなかったりしても、DBA ユーザとしてデータベースにログインできます。


```
GRANT INTEGRATED LOGIN
TO mywindowsusers
AS USER DBA
```

「Windows ユーザ・グループ用の統合化ログインの作成」 98 ページを参照してください。

統合化ログイン・パーミッションの取り消し

統合化ログイン・マッピングを削除するには、Sybase Central または Interactive SQL のいずれかを使用します。

♦ 統合化ログイン・パーミッションを取り消すには、次の手順に従います (Sybase Central の場合)。

1. DBA ユーザとしてデータベースに接続します。
2. [ログイン・マッピング] フォルダを開きます。
3. 右ウィンドウ枠で、削除するログイン・マッピングを選択し、[編集] – [削除] を選択します。

♦ 統合化ログイン・パーミッションを取り消すには、次の手順に従います (SQL の場合)。

1. DBA ユーザとしてデータベースに接続します。
2. REVOKE INTEGRATED LOGIN FROM 文を実行します。

例

次の SQL 文は、Windows ユーザの pchin から統合化ログイン・パーミッションを削除します。

```
REVOKE INTEGRATED LOGIN
FROM pchin
```

「REVOKE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

クライアント・アプリケーションからの接続

クライアント・アプリケーションをデータベースに接続するには、統合化ログインを次のいずれかの方法で使用します。

- 接続パラメータ・リストで Integrated (INT) パラメータに YES を設定します。
- 接続文字列または [接続] ダイアログでは、ユーザ ID もパスワードも指定しません。

接続文字列で Integrated=YES を指定すると、統合化ログインが試行されます。接続が失敗し、login_mode データベース・オプションに Standard,Integrated が設定されている場合は、サーバは標準ログインを試行します。「login_mode オプション [データベース]」 459 ページを参照してください。

ユーザ ID やパスワードを指定しないでデータベースに接続する場合、統合化ログインが試行されます。接続の成否は、現在のユーザ・プロファイル名がデータベース内の統合化ログイン・マッピングに一致するかどうかによって決まります。

Interactive SQL の例

たとえば、次の Interactive SQL 文を使用した接続は成功します。ただし、接続が成功するためには、ユーザは、サーバのデフォルト・データベース内の統合化ログイン・マッピングと一致するユーザ・プロファイル名を使用してログオンしていることが必要です。

```
CONNECT USING 'INTEGRATED=yes'
```

Interactive SQL 文の CONNECT は、次のすべてが true の場合にデータベースに接続できます。

- サーバが現在実行中である。
- デフォルト・データベースの login_mode データベース・オプションは、統合化ログイン接続を受け入れるよう設定されている。
- 現在のユーザのユーザ・プロファイル名と一致する統合化ログイン・マッピングまたはユーザが所属する Windows ユーザ・グループの統合化ログイン・マッピングが作成されている。
- サーバから接続についての詳細情報を要求された場合 (Interactive SQL を使用した場合など) に、情報を追加*せずに* [OK] をクリックする。

ネットワークから見た統合化ログイン

データベースがネットワーク・サーバにある場合、統合化ログインを使用するには次の 2 つの条件のどちらかが満たされていなければなりません。

- 統合化ログイン接続に使用するユーザ・プロファイルが、ローカル・コンピュータとサーバの両方に存在する必要があります。両方のコンピュータで、ユーザ・プロファイル名が同じであると同時に、ユーザ・プロファイルのパスワードも同じである必要があります。

  たとえば、ユーザ jsmith が統合化ログインを使用してネットワーク・サーバ上にロードされているデータベースに接続しようとした場合、ローカル・コンピュータとデータベース・サーバを実行しているコンピュータとの両方に同じユーザ・プロファイル名とパスワードが存在している必要があります。ユーザ jsmith は両方のコンピュータにログオンできる必要があります。

- ネットワーク・アクセスが Microsoft Domain によって制御されている場合、統合化ログインを試みるユーザは、Domain Controller サーバのドメイン・パーミッションを持っていて、ネットワークにログインしている必要があります。ローカル・コンピュータのユーザ・プロファイルと一致するネットワーク・サーバのユーザ・プロファイルは不要です。

デフォルトの統合化ログイン・ユーザの作成

デフォルトの統合化ログイン・ユーザ ID を作成すると、現在使用されているユーザ・プロファイル用の統合化ログイン・マッピングが存在しなくても、統合化ログインを介した接続が成功します。

たとえば、ユーザ・プロファイル名 JSMITH に統合化ログイン・マッピングが存在しない場合、使用されているユーザ・プロファイルが JSMITH であれば、統合化ログイン接続をしようとしても通常は失敗します。

ただし、Guest という名前のユーザ ID をデータベースに作成すると、ユーザ・プロファイル JSMITH を明示的に識別する統合化ログイン・マッピングがない場合でも、統合化ログインは Guest ユーザ ID に正常にマッピングします。

警告
デフォルト統合化ログイン・ユーザは、データベースに Guest というユーザ ID が含まれている場合は、統合化ログインを試みるすべてのユーザのデータベースへの接続を許可します。Guest ユーザ ID に対して付与されている権限が、新しく接続したユーザに対して付与されるパーミッションと権限を決定します。

セキュリティについての考慮事項：無制限データベース・アクセス

統合化ログイン機能は、SQL Anywhere セキュリティ・システムの代わりに Windows のログイン制御システムを使用しても動作し、ユーザ ID とパスワードを指定せずにデータベースに接続できます。本質的に、データベースを実行しているコンピュータにログインできるユーザは、データベース・セキュリティを通過します。

ユーザが dsmith として Windows サーバに正常にログインすると、統合化ログイン・マッピングかデフォルトの統合化ログイン・ユーザ ID のどちらかがあれば、ID をさらに確認しなくても、データベースに接続できます。

統合化ログインを使用する場合、データベース管理者は、データベースへのアクセスを制限するために、Windows によって使用されるログイン・セキュリティに特に注意する必要があります。

警告
ユーザ・プロファイル Guest を有効にしておくと、そのサーバが実行するデータベースには無制限アクセスが許可されます。

Guest ユーザ・プロファイルが有効で、かつパスワードがブランクの場合、そのサーバに対するログインはすべて成功します。ユーザ・プロファイルがそのサーバに存在することや、指定されたログイン ID にドメイン・ログイン・パーミッションがあることは要求されません。事実上、何らかのログイン ID とパスワードを使用すれば、すべてのユーザがサーバにログインできます。デフォルトでは、Guest ユーザ・プロファイルにログインします。

統合化ログイン機能を有効にしてデータベースに接続する場合は、このことに注意する必要があります。

次の例では、データベースを実行している Windows サーバに、ブランクのパスワードで有効になる Guest ユーザ・プロファイルがあることが前提となっています。

- ユーザ fran_whitney とデータベース・ユーザ ID DBA との間に、統合化ログイン・マッピングが存在します。ユーザ fran_whitney が正しいログイン ID とパスワードを使用してサーバに接続すると、完全な管理権限を持つユーザ DBA としてデータベースに接続されます。

  しかし、fran_whitney としてサーバに接続しようとした他のユーザも、指定したパスワードに関係なくサーバにログインできます。これは、Windows がデフォルトでは Guest ユーザ・プロファイルに接続しようとするためです。ログイン ID fran_whitney を使用してサーバへのログインに成功すると、権限のないユーザでも統合化ログイン・マッピングを使用する DBA としてデータベースに接続されます。

セキュリティ確保のため Guest ユーザ・プロファイルを無効にする
統合化ログインの最も安全な方法は、SQL Anywhere データベースを実行するすべての Windows コンピュータで、Guest ユーザ・プロファイルを無効にすることです。これには、Windows のユーザー マネージャ・ユーティリティを使用します。

Kerberos 認証の使用

Kerberos ログイン機能を使用すると、データベース接続とオペレーティング・システムやネットワークのログインの両方を、単一のユーザ ID とパスワードで管理できます。この項では、Kerberos ログイン機能について説明します。

Kerberos ログインの利点

Kerberos は、シークレット・キー暗号方式を使用して強力な認証と暗号化を実現するネットワーク認証プロトコルです。SQL Anywhere では、Windows 統合化ログインと同じように Kerberos を認証に使用できます。Kerberos にログインしているユーザは、ユーザ ID やパスワードを指定しなくてもデータベースに接続できます。

Kerberos を認証システムとして使用するには、認証を Kerberos に委任するよう SQL Anywhere を設定する必要があります。データベースは、Kerberos ログインを使用するよう設定し、コンピュータやネットワークへのログインに使用するユーザと、データベース・ユーザとをマッピングする必要があります。

Kerberos が設定済みの場合は、この認証メカニズムを使用してデータベースに接続しようとするユーザを認証できます。

Kerberos ログインの使用は、ユーザにとって便利であると同時に、1 つのセキュリティ・システムでデータベースとネットワークの両方のセキュリティを維持できます。次のような利点があります。

- ユーザはデータベースに接続するときにユーザ ID やパスワードを入力しなくてよい
- 複数のユーザを 1 つのデータベース・ユーザ ID にマッピングできる
- Kerberos へのログインに使用する名前とパスワードはデータベース・ユーザの ID とパスワードと一致している必要がない

警告
Kerberos ログインにはセキュリティ・システムが単一であるという利便性がありますが、そのためには、データベース管理者がセキュリティ上の重要事項を熟知しておく必要があります。「セキュリティについての考慮事項：コピーされたデータベース・ファイル」 119 ページを参照してください。

SQL Anywhere に Kerberos ソフトウェアは付属していません。Kerberos ソフトウェアは別途入手する必要があります。Kerberos ソフトウェアは、以下のコンポーネントで構成されています。

- **Kerberos ライブラリ**　Kerberos クライアントまたは GSS (Generic Security Services)-API ランタイム・ライブラリと呼ばれています。Kerberos ライブラリは、明確に定義されている GSS-API を実装したもので、Kerberos を使用するクライアント・コンピュータおよびサーバ・コンピュータでそれぞれ必要です。KDC として Active Directory を使用する場合は、サードパーティ製 Kerberos クライアント・ライブラリの代わりに、組み込みの Windows SSPI インタフェースを使用できます。

- **Kerberos キー配布センター (KDC) サーバ**　KDC はユーザおよびサーバの保管場所として機能します。また、ユーザやサーバの ID を検証します。KDC は、通常、アプリケーションやユーザ・ログインに使用しないサーバ・コンピュータにインストールされます。

SQL Anywhere は、DBLib クライアント、ODBC クライアント、OLE DB クライアント、ADO.NET クライアント、Sybase Open Client、jConnect クライアントからの Kerberos 認証をサポートします。Kerberos 認証は SQL Anywhere トランスポート・レイヤ・セキュリティ暗号化と組み合わせて使用できますが、SQL Anywhere はネットワーク通信での Kerberos 暗号化をサポートしていません。

Windows では、Kerberos を Windows ドメインとドメイン・アカウントに使用します。Active Directory Windows Domain Controllers は Kerberos KDC を実装します。この環境で認証を行うには、データベース・サーバ・コンピュータにサードパーティ製 Kerberos クライアントまたはランタイムが必要ですが、Windows クライアント・コンピュータはサードパーティ製 Kerberos クライアントまたはランタイムの代わりに組み込み Windows SSPI インタフェースを使用できます。「Windows で Kerberos ログインに SSPI を使用する」 114 ページを参照してください。

Kerberos クライアント

Kerberos 認証は、32 ビットの Windows と Linux で利用できます。テスト済みの Kerberos クライアントのリストについては、「SQL Anywhere Supported Kerberos Clients」を参照してください。

サポートされている Kerberos クライアントが使用する keytab ファイルと GSS-API ファイルのデフォルトの名前とロケーションのリストを次の表に示します。

Kerberos クライアント	デフォルト keytab ファイル	GSS-API ライブラリ・ファイル名	説明
Windows MIT Kerberos クライアント	*C:¥WINDOWS¥krb5kt*	*gssapi32.dll*	KRB5_KTNAME 環境変数を設定してからデータベース・サーバを起動することで、別の keytab ファイルを指定できます。
Windows CyberSafe Kerberos クライアント	*C:¥Program Files¥CyberSafe¥v5srvtab*	*gssapi32.dll*	CSFC5KTNAME 環境変数を設定してからデータベース・サーバを起動することで、別の keytab ファイルを指定できます。
UNIX MIT Kerberos クライアント	*/etc/krb5.keytab*	*libgssapi_krb5.so*[1]	KRB5_KTNAME 環境変数を設定してからデータベース・サーバを起動することで、別の keytab ファイルを指定できます。
UNIX CyberSafe Kerberos クライアント	*/krb5/v5srvtab*	*libgss.so*[1]	CSFC5KTNAME 環境変数を設定してからデータベース・サーバを起動することで、別の keytab ファイルを指定できます。

Kerberos クライアント	デフォルト keytab ファイル	GSS-API ライブラリ・ファイル名	説明
UNIX Heimdal Kerberos クライアント	*/etc/krb5.keytab*	*libgssapi.so.1*[1]	

[1] ファイル名はオペレーティング・システムと Kerberos クライアント・バージョンによって異なります。

Kerberos 認証の設定

次の手順では、Kerberos 認証を SQL Anywhere と組み合わせて設定および使用する方法について説明します。

♦ SQL Anywhere データベース用の Kerberos 認証の設定

1. クライアントとサーバの両方のコンピュータに、Kerberos クライアント・ソフトウェアを GSS-API ランタイム・ライブラリも含めてインストールし、設定します(現在 Kerberos を使用している場合は、この手順は実行済みです)。

   Active Directory KDC を使用している Windows クライアントでは SSPI を使用でき、Kerberos クライアントのインストールは不要です。「Windows で Kerberos ログインに SSPI を使用する」 114 ページを参照してください。

2. 必要に応じて、Kerberos プリンシパルを各ユーザの Kerberos キー配布センター (KDC：Key Distribution Center) に作成します。

   Kerberos プリンシパルとは Kerberos ユーザ ID であり、形式は *userIinstance@REALM* です。*I instance* はオプションです。Kerberos をすでに使用している場合は、プリンシパルはすでに存在しているので、各ユーザに Kerberos プリンシパルを作成する必要はありません。

   プリンシパルでは大文字と小文字が区別されるので、間違えないよう入力してください。大文字と小文字の違いしかない複数のプリンシパルのマッピングはサポートされていません (たとえば、jjordan@MYREALM.COM と JJordan@MYREALM.COM の両方でマッピングすることはできません)。

3. Kerberos プリンシパルを SQL Anywhere データベース・サーバの KDC に作成します。

   データベース・サーバ用の Kerberos プリンシパルの形式は server-name@REALM です。server-name は SQL Anywhere データベース・サーバ名です。プリンシパルでは大文字と小文字が区別されます。また、server-name ではマルチバイト文字や /、¥、@ は使用できません。以降の手順では、Kerberos プリンシパルが my_server_princ@MYREALM.COM であることを前提としています。

   ユーザと同様、サーバもみずからを KDC で認証する必要があります。このため、サーバ・サービス・プリンシパルを KDC に作成する必要があります。サーバはみずからを keytab ファイルを使用して認証します。keytab ファイルは、サーバが KDC にみずからを証明するために使用する保護された暗号化ファイルです。

4. プリンシパル *server-name@REALM* 用の keytab ファイルを、KDC から SQL Anywhere データベース・サーバを実行中のコンピュータに、セキュリティに十分注意して抽出およびコピーします。keytab ファイルのデフォルト・ロケーションは、Kerberos クライアントとプラットフォームによって異なります。keytab ファイルのパーミッションは、SQL Anywhere サーバが読み取ることができ、不正なユーザに読み取りパーミッションがないよう設定する必要があります。

Kerberos をインストールして設定したら、SQL Anywhere データベースの Kerberos ログインを有効にする必要があります。Kerberos ログインを許可するかどうかは、login_mode データベース・オプションで指定します。データベース・オプションは、それが指定されているデータベースにしか適用されないため、同じサーバ内にロードされて動作している場合でも、データベースが異なれば Kerberos ログインの設定も異なります。

login_mode データベース・オプションには、次の値を 1 つまたは複数指定できます。

- **Standard**　標準ログインを許可します。これがデフォルト設定です。標準ログインでは、ユーザ ID とパスワードの両方を入力する必要があり、統合化接続や Kerberos 接続のパラメータは使用されません。
- **Integrated**　統合化ログインを許可します。
- **Kerberos**　Kerberos ログインを許可します。

警告
login_mode データベース・オプションを Kerberos に設定すると、接続できるのは、Kerberos ログイン・マッピングを付与されているユーザだけに制限されます。このとき、ユーザ ID とパスワードを使用して接続しようとすると、エラーが発生します。ただし、DBA 権限 (完全な管理権) を持つユーザは例外です。

以降の手順は、上述の手順を実行して Kerberos が設定済みであることを前提としています。

♦ Kerberos を使用するよう SQL Anywhere を設定するには、次の手順に従います。

1. SQL Anywhere サーバを -krb または -kr オプションを使用して起動し、Kerberos 認証を有効にします (代わりに -kl オプションを使用して、GSS-API ライブラリのロケーションを指定し、Kerberos を有効にすることもできます)。

   次の項を参照してください。

   - 「-kl サーバ・オプション」 178 ページ
   - 「-kr サーバ・オプション」 178 ページ
   - 「-krb サーバ・オプション」 179 ページ

   次のコマンドは、Kerberos プリンシパル my_server_princ@MYREALM.COM のデータベース・サーバを起動します。

   ```
   dbsrv10 -krb -n my_server_princ C:¥kerberos.db
   ```


2. パブリック・オプションまたは一時的なパブリック・オプション login_mode を Kerberos を含む値に変更します。このオプションの設定を変更するには DBA 権限が必要です。「login_mode オプション [データベース]」 459 ページを参照してください。

   ```
   SET OPTION PUBLIC.login_mode = 'Kerberos,Standard'
   ```

3. クライアントが使用するデータベース・ユーザ ID を作成します。既存のデータベース・ユーザに適切なパーミッションがあれば、その ID を Kerberos ログインに使用できます。

   ```
   GRANT CONNECT TO "kerberos-user"
   IDENTIFIED BY "abc123"
   ```

4. クライアントの Kerberos プリンシパルから既存のデータベース・ユーザ ID へのマッピングを作成します。作成するには、GRANT KERBEROS LOGIN TO 文を実行します (この文の実行には DBA 権限が必要)。「GRANT 文」 『SQL Anywhere サーバ - SQL リファレンス』と「Kerberos ログイン・マッピングの作成」 113 ページを参照してください。

   次に例を示します。

   ```
   GRANT KERBEROS LOGIN TO "pchin@MYREALM.COM"
   AS USER "kerberos-user"
   ```

   マッピングのない Kerberos プリンシパルが使用されているときに接続する場合は、Guest データベース・ユーザ ID が存在し、パスワードが設定されていることを確認します。「デフォルトの統合化ログイン・ユーザの作成」 105 ページを参照してください。

5. クライアント・ユーザが Kerberos プリンシパルを使用してログオン済みである (有効な Kerberos TGT：Ticket Granting Ticket がある) こと、およびクライアントの Kerberos チケットの期限が切れていないことを確認します。ドメイン・アカウントにログインしている Windows ユーザは、TGT をすでに持っており、プリンシパルに必要なパーミッションがあれば、サーバに認証されます。

   TGT はユーザ・パスワードを使用して暗号化された Kerberos チケットで、ユーザ ID の検証にチケット保証サービス (TGS：Ticket Granting Service) が使用します。

6. KERBEROS 接続パラメータ (ほとんどの場合は KERBEROS=YES、ただし KERBEROS=SSPI または KERBEROS=*GSS-API-library-file* も使用可) を使用して、クライアントから接続します。ユーザ ID またはパスワードの接続パラメータが指定された場合は、無視されます。次に例を示します。

   ```
   dbisql -c "KERBEROS=YES;ENG=my_server_princ"
   ```

Interactive SQL の例

たとえば、次の Interactive SQL 文を使用した接続は成功します。ただし、接続が成功するためには、ユーザは、サーバのデフォルト・データベース内の Kerberos ログイン・マッピングと一致するユーザ・プロファイル名を使用してログオンしていることが必要です。

```
CONNECT USING 'KERBEROS=YES'
```

Interactive SQL 文の CONNECT は、以下のすべてが true の場合にデータベースに接続できます。

- サーバが現在実行中である。

- 現在のサーバのデフォルト・データベースが、Kerberos 認証接続を受け入れることができる。
- Kerberos ログイン・マッピングがユーザの現在の Kerberos プリンシパルに対して作成されている。
- サーバから接続についての詳細情報を要求された場合 (Interactive SQL を使用した場合など) に、情報を追加*せずに* [OK] をクリックする。

Open Client 接続と jConnect 接続

Open Client アプリケーションまたは jConnect アプリケーションから接続するには、前述の手順 (「Kerberos を使用するよう SQL Anywhere を設定するには、次の手順に従います。」で説明している手順) に従い、Open Client/jConnect で Kerberos 認証を Adaptive Server Enterprise と組み合わせて使用するよう設定します。サーバ名は SQL Anywhere サーバ名である必要があります。名前の大文字と小文字は区別されます。Open Client または jConnect の代替サーバ名を接続に使用することはできません。

Kerberos プリンシパルの設定と keytab の抽出の詳細については、http://www.sybase.com/detail?id=1029260 を参照してください。

参照

- 「-krb サーバ・オプション」 179 ページ
- 「-kr サーバ・オプション」 178 ページ
- 「-kl サーバ・オプション」 178 ページ
- 「login_mode オプション [データベース]」 459 ページ
- 「GRANT 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「Kerberos 接続パラメータ [KRB]」 253 ページ
- 「Kerberos 接続のトラブルシューティング」 115 ページ

Kerberos ログイン・マッピングの作成

Kerberos ログイン・マッピングを作成するには、Sybase Central または SQL 文のいずれかを使用します。

♦ Kerberos ログイン・マッピングを作成するには、次の手順に従います (Sybase Central の場合)。

1. DBA ユーザとしてデータベースに接続します。
2. [ログイン・マッピング] フォルダを開きます。
3. [ファイル] メニューから、[新規] – [ログイン・マッピング] を選択します。

   [ログイン・マッピングの作成] ウィザードが表示されます。

♦ Kerberos ログイン・マッピングを作成するには、次の手順に従います (SQL の場合)。

1. DBA ユーザとしてデータベースに接続します。

2. GRANT KERBEROS LOGIN TO 文を実行します。

   「GRANT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

例

次の SQL 文は、Windows ユーザの pchin に KERBEROS ログイン・パーミッションを付与します。

```
GRANT KERBEROS LOGIN TO "pchin@MYREALM.COM"
AS USER "kerberos-user"
```

Kerberos ログイン・パーミッションの取り消し

Kerberos ログイン・マッピングを削除するには、Sybase Central または SQL 文のいずれかを使用します。

♦ Kerberos ログイン・マッピングを取り消すには、次の手順に従います (Sybase Central の場合)。

1. DBA ユーザとしてデータベースに接続します。
2. [ログイン・マッピング] フォルダを開きます。
3. 右ウィンドウ枠で、削除するログイン・マッピングを選択し、[編集] – [削除] を選択します。

♦ Kerberos ログイン・マッピングを取り消すには、次の手順に従います (SQL の場合)。

1. DBA ユーザとしてデータベースに接続します。
2. REVOKE KERBEROS LOGIN FROM 文を実行します。

   「REVOKE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

例

次の SQL 文は、Windows ユーザの pchin から KERBEROS ログイン・パーミッションを削除します。

```
REVOKE KERBEROS LOGIN
FROM "pchin@MYREALM.COM"
```

Windows で Kerberos ログインに SSPI を使用する

Windows ドメインを使用している Windows クライアント・コンピュータでは SSPI を使用でき、Kerberos クライアントをクライアント・コンピュータにインストールする必要はありません。Windows ドメイン・アカウントには、それに関連付けられた Kerberos プリンシパルがあらかじめ用意されています。たとえば、myrealm.com Windows ドメインのアカウント pchin は、pchin@MYREALM.COM Kerberos プリンシパルにあらかじめ関連付けられているのが一般的で

す。この方法で認証するクライアントでは、Kerberos (KRB) 接続パラメータを SSPI に設定する必要があります。

接続文字列に Kerberos=SSPI と指定されている場合、Kerberos ログインが試行されます。

次の手順は、Kerberos 認証が設定済みであることを前提としています。「Kerberos 認証の設定」 110 ページを参照してください。

♦ **SSPI を使用して接続するには、次の手順に従います。**

1. SQL Anywhere サーバを -krb オプションを指定して起動し、Kerberos 認証を有効にします。

   ```
   dbeng10 -krb -n my_server_princ C:¥kerberos.db
   ```

2. パブリック・オプションまたは一時的なパブリック・オプション login_mode を Kerberos を含む値に変更します。このオプションの設定を変更するには DBA 権限が必要です。

   ```
   SET OPTION PUBLIC.login_mode = 'Kerberos'
   ```

3. クライアントが使用するデータベース・ユーザ ID を作成します。既存のデータベース・ユーザに適切なパーミッションがあれば、その ID を Kerberos ログインに使用できます。

   ```
   GRANT CONNECT TO "kerberos-user"
   IDENTIFIED BY "abc123"
   ```

4. クライアントの Kerberos プリンシパルから既存のデータベース・ユーザ ID へのマッピングを作成します。作成するには、GRANT KERBEROS LOGIN TO 文を実行します (この文の実行には DBA 権限が必要)。

   ```
   GRANT KERBEROS LOGIN TO "pchin@MYREALM.COM"
   AS USER "kerberos-user"
   ```

5. クライアント・コンピュータからデータベースに接続します。

   ```
   dbisql -c "KERBEROS=SSPI;ENG=my_server_princ"
   ```

   次の Interactive SQL 文を使用した接続も成功します。ただし、接続が成功するためには、ユーザは、サーバのデフォルト・データベース内の Kerberos ログイン・マッピングと一致するユーザ・プロファイル名を使用してログオンしていることが必要です。

   ```
   CONNECT USING 'KERBEROS=SSPI'
   ```

Kerberos 接続のトラブルシューティング

Kerberos 認証を有効にしたり使おうとしたときに予期しないエラーが発生した場合は、データベース・サーバとクライアントの両方で追加の診断メッセージを有効にすることをおすすめします。

データベース・サーバの起動時に -z オプションを指定します。または、すでに実行中のサーバのサーバ・コンソールに追加診断メッセージを表示するには CALL sa_server_option ('DebuggingInformation', 'ON') を使用します。LogFile 接続パラメータを使用すると、指定したファイルにクライアント診断メッセージが書き込まれます。LogFile 接続パラメータを使用する代わりに、コマンド dbping -z を実行することもできます。-z パラメータにより診断メッセージが表示され、接続の問題の原因を特定するのに役立ちます。

データベース・サーバの起動に関する問題

現象	一般的な解決策
「Kerberos GSS-API ライブラリをロードできません。」メッセージ	◆ Kerberos クライアントが、GSS-API ライブラリも含めて、データベース・サーバ・マシンにインストールされていることを確認します。 ◆ ロードしようとしたライブラリ名が、データベース・サーバの -z 出力にリストされます。ライブラリ名が正しいことを確認し、必要に応じて -kl オプションを使用して正しいライブラリ名を指定します。 ◆ サポートしているライブラリがあるディレクトリがライブラリ・パス (Windows では %PATH%) にリストされていることを確認します。 ◆ GSS-API ライブラリにエントリ・ポイントがないことがデータベース・サーバの -z 出力に示されていた場合、そのライブラリはサポートされている Kerberos Version 5 GSS-API ライブラリではありません。
「サーバ名 "*server-name*" の Kerberos クレデンシャルを取得できません。」メッセージ	◆ *server-name@REALM* のプリンシパルが KDC にあることを確認します。プリンシパルは大文字と小文字が区別されるので、データベース・サーバ名の大文字と小文字がプリンシパル名のユーザ部分の大文字と小文字と一致していることを確認します。 ◆ SQL Anywhere サーバ名がプリンシパルのプライマリ/ユーザ部分になっていることを確認します。 ◆ サーバのプリンシパルが keytab ファイルに抽出されていること、およびその keytab ファイルが Kerberos クライアントに対して適切なロケーションにあることを確認します。「Kerberos クライアント」 109 ページを参照してください。 ◆ データベース・サーバ・マシン上の Kerberos クライアントの領域がサーバ・プリンシパルの領域と異なっている場合は、-kr オプションを使用してサーバ・プリンシパルの領域を指定します。
「Kerberos ログインが失敗しました。」クライアント・エラー	◆ データベース・サーバの診断メッセージを確認します。サーバが使用する keytab ファイルに関する問題のなかには、クライアントが認証しようとするまで検出されないものがあります。

Kerberos クライアント接続のトラブルシューティング

クライアントが Kerberos 認証を使用して接続しようとしてエラーが発生した場合について、次の表に示します。

現象	一般的な解決策
「Kerberos ログインはサポートされていません。」エラーが発生し、LogFile にメッセージ 「Kerberos GSS-API ライブラリのロードに失敗しました」が出力されている。	◆ Kerberos クライアントが、GSS-API ライブラリも含めて、クライアント・コンピュータにインストールされていることを確認します。 ◆ LogFile に指定したファイルに、ロードしようとしたライブラリがリストされています。ライブラリ名が正しいことを確認し、必要があれば Kerberos 接続パラメータを使用して正しいライブラリ名を指定します。 ◆ サポートしているライブラリがあるディレクトリがライブラリ・パス (Windows では %PATH%) にリストされていることを確認します。 ◆ GSS-API ライブラリにエントリ・ポイントがないことが LogFile 出力に示されていた場合、そのライブラリはサポートされている Kerberos Version 5 GSS-API ライブラリではありません。
「Kerberos ログインはサポートされていません。」エラー	◆ データベース・サーバに対して -krb、-kl、-kr の各サーバ・オプションが 1 つ以上指定されており、Kerberos ログインが有効になっていることを確認します。 ◆ Kerberos がクライアントとサーバの両プラットフォーム上の SQL Anywhere でサポートされていることを確認します。
「Kerberos ログインが失敗しました。」エラー	◆ ユーザが Kerberos にログイン済みであること、およびそのユーザに期限が切れていない有効な TGT があることを確認します。 ◆ クライアント・コンピュータとサーバ・コンピュータとの間で、時刻のずれが 5 分未満であることを確認します。
「ログイン・モード 'Kerberos' は、login_mode 設定で許可されていません。」エラー	Kerberos ログインが許可されるには、login_mode オプションのパブリックまたは一時的なパブリックのデータベース・オプションの設定に値 Kerberos が含まれている必要があります。

現象	一般的な解決策
「ログイン ID '*client-Kerberos-principal*' はどのデータベース・ユーザ ID にもマップされていません。」	◆ Kerberos プリンシパルが GRANT KERBEROS LOGIN 文を使用してデータベース・ユーザ ID にマッピングされている必要があります。GRANT KERBEROS LOGIN 文には完全なクライアント・プリンシパルが領域を含めて指定されている必要があります。また、インスタンスまたは領域しか違わないプリンシパルも別のプリンシパルとして扱われます。 ◆ また、明示的にマッピングされていない有効な Kerberos プリンシパルを接続可能にするには、GRANT CONNECT を使用して、ゲスト・データベース・ユーザ ID とパスワードを作成します。

セキュリティについての考慮事項：一時的にパブリック・オプションを設定してセキュリティを追加する

標準ログイン、統合化ログイン、Kerberos ログインの組み合わせを許可するよう、SET OPTION 文を使用してデータベースの login_mode オプションの値を設定すると、指定したログイン接続がそのデータベースで永続的に有効になります。たとえば、次の文は標準ログインと統合化ログインを永続的に許可します。

```
SET OPTION PUBLIC.login_mode = 'Standard,Integrated'
```

データベースを停止して再起動した場合でも、このオプションの値は変わらず、統合化ログインも有効のままです。

login_mode オプションを SET TEMPORARY OPTION を使用して設定した場合、統合化ログインによるユーザ・アクセスは可能になりますが、データベースがシャットダウンされるまでに限られます。次の文は、オプションの値を一時的に変更します。

```
SET TEMPORARY OPTION PUBLIC.login_mode = 'Standard,Integrated'
```

永久オプション値が Standard の場合、データベースは停止時にその値に戻ります。

一時的なパブリック・オプションの設定は、データベース・アクセスの追加のセキュリティ対策と見ることができます。統合化ログインまたは Kerberos ログインを有効にすると、データベースはそれ自体が動作しているオペレーティング・システムのセキュリティに依存するためです。データベースが停止されて別のコンピュータ (ユーザのコンピュータなど) にコピーされると、データベースへのアクセスには SQL Anywhere のセキュリティ・モデルが使用され、データベースがコピーされたコンピュータのオペレーティング・システムのセキュリティ・モデルは使用されなくなります。

参照

- 「セキュリティについての考慮事項：コピーされたデータベース・ファイル」 119 ページ
- 「SET OPTION 文」『SQL Anywhere サーバ - SQL リファレンス』

セキュリティについての考慮事項：コピーされたデータベース・ファイル

データベース・ファイルがコピー可能な場合、一時的なパブリック login_mode オプションを使用する必要があります (統合化ログインと Kerberos ログインの両方で)。ファイルをコピーする場合、統合化ログインと Kerberos ログインはデフォルトではサポートされません。

不正アクセスから保護する必要がある機密情報がデータベースに含まれている場合、データベース・ファイルが格納されているコンピュータを不正アクセスから保護する必要があります。保護できない場合、データベース・ファイルがコピー可能であり、別なコンピュータ上でデータに不正アクセスできることから、セキュリティ・リスクとなります。このような環境でセキュリティを向上させるには、次の手順に従うことをおすすめします。

- ユーザ・パスワード、特に DBA 権限のユーザ・パスワードを複雑にして、推測しにくくします。
- PUBLIC.login_mode データベース・オプションを Standard に設定します。統合化ログインまたは Kerberos ログインを有効にする場合は、サーバを起動するたびに変更されるパブリック・オプションを *temporary* だけにします。これにより、データベースがコピーされたとしても、可能になるのは標準ログインだけになります。「セキュリティについての考慮事項：一時的にパブリック・オプションを設定してセキュリティを追加する」 118 ページを参照してください。
- データベース・ファイルを AES 暗号化アルゴリズムを使用して強力に暗号化します。暗号化キーは複雑にして、推測しにくくします。

第 4 章

クライアント／サーバ通信

目次

サポートされているネットワーク・プロトコル

適切に設定された SQL Anywhere データベース・サーバは、次のネットワークとプロトコル上で実行できます。

- **Windows (Windows 2003 Server を除く)** TCP/IP または SPX プロトコル
- **Windows 2003 Server (32 ビット)** TCP/IP または SPX プロトコル
- **Windows 2003 Server (64 ビット)** TCP/IP プロトコル
- **Windows CE** TCP/IP プロトコル
- **NetWare** TCP/IP または SPX プロトコル
- **UNIX** TCP/IP プロトコル

各プラットフォームのクライアント・ライブラリは、対応するサーバと同じプロトコルをサポートします。SQL Anywhere を問題なく実行するには、クライアント・コンピュータとサーバ・コンピュータの両方に、ネットワーク・プロトコル (TCP/IP か SPX、またはその両方) をインストールし、正しく設定してください。

TCP/IP プロトコルの使用

TCP/IP は、Internet と WWW の普及に伴い広く使用されるようになったプロトコル・スイートです。

UDP は、トランスポート・レイヤのプロトコルです。これは IP の最上部のプロトコルです。SQL Anywhere では、初期サーバ名解析には IP の上で UDP を使用し、その後の接続と通信には TCP を使用します。

TCP/IP プロトコルを使用するときは、トランスポート・レイヤ・セキュリティと ECC または RSA 暗号化テクノロジを使用して、クライアント／サーバ通信を安全化できます。

「トランスポート・レイヤ・セキュリティ」 949 ページを参照してください。

SQL Anywhere での IPv6 サポート

IPv6 対応のコンピュータでは、ネットワーク・データベース・サーバは、コンピュータの IPv4 アドレスの他に、デフォルトですべての IPv6 アドレスを受信します。IPv6 は Windows、Linux、Mac OS X、Solaris、AIX、HP-UX でサポートされています。

ほとんどの場合、IPv6 を使用するためにサーバの StartLine を変更する必要はありません。IP アドレスの指定が必要な場合は、サーバ・ライブラリとクライアント・ライブラリがどちらも IPv4 アドレスと IPv6 アドレスを受け付けます。たとえば、コンピュータで複数のネットワーク・カードを使用できる場合、IPv4 アドレスと IPv6 アドレスが複数存在することがあります。データベース・サーバが受信する IPv6 アドレスを 1 つに制限する場合は、アドレスを次のフォーマットで指定できます。

```
dbsrv10 -x tcpip(MyIP=fd77:55f:5a64:52a:202:5445:5245:444f) ...
```

同様に、クライアント・アプリケーションでサーバの IP アドレスを指定する必要がある場合は、接続文字列または DSN に次のフォーマットでアドレスを指定できます。

```
...;LINKS=tcpip(HOST=fe80::5445:5245:444f);...
```

各インタフェースにインタフェース識別子が与えられており、IPv6 アドレスの末尾に示されます。たとえば、*ipconfig.exe* にアドレス fe80::5445:5245:444f%7 がリストされた場合、インタフェース識別子は 7 です。IPv6 アドレスを Windows プラットフォームに指定する場合は、インタフェース識別子を使用する必要があります。UNIX では、インタフェース識別子とインタフェース名のどちらでも指定できます (インタフェース名は、ifconfig によってレポートされるインタフェース識別子の名前です)。たとえば、IPv6 アドレスが fe80::5445:5245:444f%eth1 である場合、インタフェース名は eth1 です。Linux (カーネル 2.6.13 以上) で IPv6 アドレスを指定するときには、インタフェース識別子が必要です。この要件は次のプロトコル・オプションによって指定された値に影響します。

- ブロードキャスト
- ホスト
- MyIP

たとえば、*ipconfig.exe* に 2 つのインタフェースがリストされており、片方の識別子が 1 でもう片方が 2 だととします。インタフェース番号 2 によって使用されているネットワーク上のデータ

ベース・サーバを検索している場合、クライアント・ライブラリに対してそのインタフェースにだけブロードキャストするよう指示できます。

```
LINKS=tcpip(BROADCAST=ff02::1%2)
```

ff02::1 は IPv6 リンク・ローカルのマルチキャスト・アドレスです。

参照

- 「Broadcast プロトコル・オプション [BCAST]」 266 ページ
- 「Host プロトコル・オプション [IP]」 272 ページ
- 「MyIP プロトコル・オプション [ME]」 281 ページ

Windows での TCP/IP の使用

すべての Windows プラットフォーム上のデータベース・サーバの TCP/IP 実装では、Winsock 2.2 を使用します。Windows CE 上のクライアントは、Winsock 1.1 standard を使用します。

TCP/IP がインストールされていない場合は、[コントロール パネル] の [ネットワーク] をダブルクリックして TCP/IP プロトコルをインストールできます。

「DLL プロトコル・オプション」 270 ページを参照してください。

TCP/IP パフォーマンスのチューニング

パケット・サイズを大きくすると、クエリの応答時間を短縮できます。特に、クライアントとサーバ・プロセスの間で大量のデータを転送するクエリでは大変有効です。パケット・サイズを設定するには、データベース・サーバのコマンドで -p オプションを使用するか、接続プロファイルに CommBufferSize (CBSIZE) 接続パラメータを設定します。

参照

- 「-p サーバ・オプション」 185 ページ
- 「CommBufferSize 接続パラメータ [CBSIZE]」 234 ページ

ファイアウォール経由の接続

クライアント・アプリケーションがファイアウォールの片側にあり、サーバがもう片側にある場合は、接続が制限されます。ファイアウォール・ソフトウェアは、ネットワーク・ポートに従って、ネットワーク・パケットをフィルタリングします。また通常は、UDP パケットはファイアウォールを通過できません。

ファイアウォール経由で接続する場合は、アプリケーションの接続文字列の CommLinks (LINKS) 接続パラメータで一連のプロトコル・オプションを使用してください。

- **Host**　このパラメータには、データベース・サーバを実行しているホスト名を設定します。IP と短縮してもかまいません。

- **ServerPort**　データベース・サーバがデフォルト・ポート 2638 を使用していない場合、使用しているポートを指定します。Port と短縮してもかまいません。
- **ClientPort**　このパラメータには、使用するクライアント・アプリケーションで有効な範囲の値を設定します。Cport と短縮してもかまいません。このオプションは、ファイアウォールの設定によっては必要でない場合があります。
- **DoBroadcast=NONE**　サーバに接続するときに UDP が使用されないようにするには、このパラメータを設定します。

ファイアウォールは、SQL Anywhere サーバのアドレスとすべての SQL Anywhere クライアントのアドレスとの間の TCP/IP トラフィックを許可するように設定します。SQL Anywhere サーバのアドレスは、SQL Anywhere サーバを実行中のコンピュータの IP アドレス (HOST パラメータ) と SQL Anywhere サーバの IP ポート番号 (ServerPort プロトコル・オプション、デフォルトは 2638) です。各 SQL Anywhere クライアントのアドレスは、クライアント・コンピュータの IP アドレスとクライアント IP ポートの範囲 (ClientPort プロトコル・オプション) で構成されます。設定を簡単にするために、すべてのクライアント・ポートを含めることができます。指定のクライアント・ポートのみを含める場合は、クライアント・ポートが再び利用されるまでに数分のタイムアウトがあるため、各クライアント・コンピュータからの同時接続の最大数よりも多いポート数で範囲を指定します。

「ClientPort プロトコル・オプション [CPORT]」 268 ページを参照してください。

例

次の接続文字列は、クライアント・アプリケーションをポート 5050 ー 5060 に制限します。また、サーバ・ポート 2020 を使用するアドレス myhost のコンピュータで実行されているサーバ myeng に接続します。DoBroadcast オプションを指定しているため、UDP ブロードキャストは実行されません。

```
ENG=myeng;LINKS=tcpip(ClientPort=5050-5060;HOST=myhost;PORT=2020;DoBroadcast=NONE)
```

参照

- 「CommLinks 接続パラメータ [LINKS]」 235 ページ
- 「ClientPort プロトコル・オプション [CPORT]」 268 ページ
- 「ServerPort プロトコル・オプション [PORT]」 283 ページ
- 「Host プロトコル・オプション [IP]」 272 ページ
- 「DoBroadcast プロトコル・オプション [DOBROAD]」 271 ページ

ダイヤルアップ・ネットワーク接続での接続

接続オプションとプロトコル・オプションを使用して、ダイヤルアップ・リンク経由でデータベースに接続できます。

クライアント側では、以下のプロトコル・オプションを指定してください。

- **Host パラメータ**　Host (IP) プロトコル・オプションを使用して、データベース・サーバのホスト名または IP アドレスを指定します。「Host プロトコル・オプション [IP]」 272 ページを参照してください。

- **DoBroadcast パラメータ**　Host (IP) プロトコル・オプションを指定した場合は、データベース・サーバでブロードキャスト検索を行う必要はありません。このため、ダイレクト・ブロードキャストを使用してください。「DoBroadcast プロトコル・オプション [DOBROAD]」 271 ページを参照してください。
- **MyIP パラメータ**　クライアント側では、**MyIP=NONE** に設定してください。「MyIP プロトコル・オプション [ME]」 281 ページを参照してください。
- **TIMEOUT パラメータ**　サーバを検索する間のクライアントの待機時間を長くするには、TIMEOUT (TO) プロトコル・オプションを設定します。「Timeout プロトコル・オプション [TO]」 286 ページを参照してください。

通常、CommLinks (LINKS) 通信パラメータは次のようになります。

```
LINKS=tcpip(MyIP=NONE;DoBroadcast=DIRECT;HOST=server_ip)
```

TCP/IP を使用したクライアント／サーバ通信の暗号化

デフォルトでは通信パケットが暗号化されないため、セキュリティに関して潜在的な危険があります。単純暗号化またはトランスポート・レイヤ・セキュリティを使用して、TCP/IP を介したクライアント・アプリケーションとデータベース・サーバ間の通信を安全化できます。トランスポート・レイヤ・セキュリティにより、サーバ認証、ECC または RSA 暗号化テクノロジを使用した強力な暗号化、およびデータ整合性を保護するその他の機能が提供されます。

「トランスポート・レイヤ・セキュリティ」 949 ページを参照してください。

LDAP サーバを使用した接続

Windows (Windows CE を除く)、UNIX、または NetWare プラットフォーム上で動作している場合、中央 LDAP サーバを指定して企業内の全データベース・サーバを追跡できます。データベース・サーバ自体を LDAP サーバに登録する場合、クライアントは LDAP サーバに問い合わせてデータベース・サーバを検索できます。この場合、サーバが WAN 上、LAN 上、またはファイアウォールの外側にあっても検索できます。クライアントは、IP アドレス (HOST=) を指定する必要はありません。サーバ列挙ユーティリティ (dblocate) も LDAP サーバを使用してそのようなサーバを検索できます。

LDAP は TCP/IP とともに使用し、ネットワーク・データベース・サーバ上でのみ使用されます。

この機能を有効にするには、LDAP サーバの検索方法と接続方法に関する情報を含むファイルをデータベース・サーバ・コンピュータと各クライアント・コンピュータに作成してください。デフォルトで、このファイル名は *saldap.ini* ですが、これは設定可能です。このファイルがない場合、LDAP のサポートは何も通知されず無効になっています。

このファイルは、LDAP パラメータに完全なパスを指定していない場合は SQL Anywhere 実行プログラム (たとえば、Windows の場合は *install-dir¥win32*) と同じディレクトリ置く必要があります。このファイルは、次のフォーマットになります。

```
[LDAP]
server=computer-running-LDAP-server
```

```
port=port-number-of-LDAP-server
basedn=Base-DN
authdn=Authentication-DN
password=password-for-authdn
search_timeout=age-of-timestamps-to-be-ignored
update_timeout=frequency-of-timestamp-updates
read_authdn=read-only-authentication-domain-name
read_password=password-for-authdn
```

ファイル非表示ユーティリティ (dbfhide) を使用して、単純暗号化によって *saldap.ini* ファイルの内容を読みにくくできます。ファイルの名前が *ldap.ini* でない場合は、LDAP パラメータを使用してファイル名を指定する必要があります。

「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。

server LDAP サーバを実行中のコンピュータの名前または IP アドレス。この値は NetWare および UNIX で必要です。このエントリが Windows にない場合、Windows がローカルのドメイン・コントローラで動作中の LDAP サーバを検索します。

port LDAP サーバで使用されるポート番号。デフォルトは 389 です。

basedn SQL Anywhere エントリが格納されているサブツリーのドメイン名。これはデフォルトでツリーのルートになります。

authdn 認証ドメイン名。ドメイン名は、LDAP ディレクトリにある既存のユーザ・オブジェクトである必要があります。このディレクトリには basedn への書き込みアクセスがあります。これはデータベース・サーバで必要ですが、クライアントでは無視されます。

password authdn のパスワード。これはデータベース・サーバで必要ですが、クライアントでは無視されます。

search_timeout タイムスタンプがクライアントまたはサーバ列挙 (dblocate) ユーティリティで無視されるタイムスタンプの経過時間。値 0 はこのオプションを無効にして、すべてのエントリが現在のものと見なされます。デフォルト値は 600 秒 (10 分) です。

update_timeout LDAP ディレクトリ内のタイムスタンプの更新頻度。値として 0 を指定するとこのオプションは無効になり、データベース・サーバはタイムスタンプを更新しません。デフォルト値は 120 秒 (2 分) です。

read_authdn 読み取り専用の認証ドメイン名。ドメイン名は、LDAP ディレクトリにある既存のユーザ・オブジェクトを指定します。このディレクトリには basedn への読み取りアクセスがあります。このパラメータが必要なのは、検索の実行前に LDAP サーバで非匿名バインドが必要な場合だけです。たとえば、Active Directory が LDAP サーバとして使用されている場合、通常はこのフィールドが必要です。このパラメータが指定されていない場合、バインドは匿名になります。

read_password authdn のパスワード。クライアントにこのパラメータが必要なのは、read_authdn パラメータが指定されている場合だけです。

例

次に、*saldap.ini* ファイルの例を示します。

```
[LDAP]
server=ldapserver
basedn=dc=iAnywhere,dc=com
```



```
authdn=cn=SAServer,ou=iAnywhereASA,dc=iAnywhere,dc=com
password=secret
```

エントリが iAnywhereASA と呼ばれる basedn のサブツリーに保管されます。このエントリは SQL Anywhere が LDAP を使用できるようになる前に作成します。サブツリーを作成するには、LDAPADD ユーティリティを使用します。次のような情報を提供します。

```
dn: ou=iAnywhereASA,basedn
objectClass: organizationalUnit
objectClass: top
ou: iAnywhereASA
```

サーバは、起動時に、LDAP ファイル内で同じ名前を持つ既存のエントリがないかどうかを確認します。そのようなエントリが見つかると、LDAP のロケーション・エントリと起動しようとしているデータベース・サーバとが一致した場合、または LDAP エントリのタイムスタンプ・フィールドが 10 分より前 (タイムアウト値は設定可能) の場合に、エントリは置き換えられます。

このいずれも該当しない場合、起動しようとしているデータベース・サーバと同じ名前の別のサーバがあることになり、起動が失敗します。

LDAP のエントリを確実に最新のものにするために、データベース・サーバは LDAP エントリ内のタイムスタンプ・フィールドを 2 分ごとに更新します。エントリのタイムスタンプが 10 分以上古い場合、クライアントは LDAP エントリを無視します。これらの設定はいずれも変更可能です。

クライアントでは、ブロードキャストを実行する前に LDAP ディレクトリが検索されるので、データベース・サーバが見つかるとブロードキャストは送信されません。LDAP 検索は非常に高速なので、失敗しても認識できるほどの遅延は発生しません。

サーバ列挙ユーティリティ (dblocate) も LDAP を使用します。LDAP にリストされているすべてのデータベース・サーバが、返されるデータベース・サーバのリストに追加されます。これにより、サーバ列挙ユーティリティ (dblocate) が、たとえばブロードキャストが到達しなかったデータベース・サーバなど、通常どおりに返されなかったサーバをリストします。10 分以上古いタイムスタンプを持つエントリは含まれません。

SPX プロトコルの使用

SPX は、Novell から提供されるプロトコルです。SQL Anywhere for NetWare と Windows では SPX プロトコルを使用できます。この項では、さまざまなオペレーティング・システム上で SPX を使用するためのヒントを説明します。

SPX は 64 ビット・オペレーティング・システムをサポートしていません。

SPX 経由での接続

一部のコンピュータでは NetWare バインダリを使用できます。該当するのは、Client Service for NetWare がインストールされた NetWare サーバまたは Windows コンピュータです。これらのコンピュータ上にあるクライアント・アプリケーションは、接続先のサーバでもバインダリを使用していれば、サーバとの接続にブロードキャストを使用する必要はありません。

バインダリを使用していないコンピュータ上で実行されているアプリケーションは、次のいずれかを使用して接続します。

- **明示的なアドレス**　HOST プロトコル・オプションを使用すると、アドレスを明示的に指定できます。「Host プロトコル・オプション [IP]」 272 ページを参照してください。
- **ブロードキャスト**　バインダリにサーバがない場合、クライアントはブロードキャストを実行してサーバを検索します。DoBroadcast (DOBROAD) プロトコル・オプションを NONE (クライアント側) または NO (サーバ側) に設定すると、ブロードキャストを無効にできます。「DoBroadcast プロトコル・オプション [DOBROAD]」 271 ページを参照してください。

パフォーマンス改善のための通信圧縮設定の調整

1 つまたはすべての接続での圧縮を有効にすると同時に、パケットを圧縮する最小のサイズを設定すると、状況によっては SQL Anywhere のパフォーマンスが大幅に向上する可能性があります。

個々の状況で圧縮を有効にすることが役立つかどうかを判別するには、該当するネットワークで該当のアプリケーションを使用してパフォーマンス分析を実行してから、運用環境で圧縮を使用することをおすすめします。パフォーマンスの成果は、使用するネットワーク、アプリケーション、転送するデータによって異なります。

圧縮をチューニングする最も基本的な方法は、接続レベルまたはサーバ・レベルで、Compression (COMP) 接続パラメータを有効または無効にするという単純な方法です。圧縮のパフォーマンスの高度な微調整には CompressionThreshold (COMPTH) 接続パラメータを使用します。

圧縮機能を有効にすると、データ・パケットに格納される情報量が増大し、特定のデータ・セットの送信に必要なパケット数が減少します。パケット数を減らすと、データを高速で送信できます。

パフォーマンス分析の詳細については、「パフォーマンス・モニタの統計値」『SQL Anywhere サーバ - SQL の使用法』または「sa_conn_compression_info システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

圧縮の有効化

次のような状況では、1 つ (またはすべて) の接続で圧縮を有効化すると、SQL Anywhere のパフォーマンスが大幅に向上する可能性があります。

- 一部の無線ネットワーク、モデム、シリアル・リンク、WAN などの低速ネットワークで使用するとき。
- 圧縮機能が組み込まれている低速ネットワークで SQL Anywhere 暗号化を使用するとき。これは、パケットが圧縮されてから暗号化されるためです。

ただし、圧縮の有効化は、パフォーマンス低下の原因になる可能性もあります。次のような例があります。

- 通信の圧縮には、より多くのメモリと CPU が使用される。このため、特に LAN やその他の高速ネットワークを使用する場合に、パフォーマンスが低下することがあります。
- ほとんどのモデムと一部の低速ネットワークに、すでに圧縮機能が組み込まれている。この場合、SQL Anywhere で通信圧縮を行っても、データの暗号化を同時に実行しないかぎり、パフォーマンスはほとんど向上しません。

圧縮の詳細については、「Compress 接続パラメータ [COMP]」 237 ページと「-pc サーバ・オプション」 185 ページを参照してください。

圧縮のスレッショルドの変更

SQL Anywhere のパフォーマンスは、圧縮のスレッショルドを調整することによっても向上させることができます。ほとんどのネットワークでは、圧縮のスレッショルドを変更する必要はありません。

圧縮が有効な場合、パケットは、各々のサイズに応じて圧縮するかどうかを決定します。たとえば、SQL Anywhere では、圧縮のスレッショルドよりも小さいパケットは、通信の圧縮が有効な場合でも圧縮されません。同様に、小さなパケット (100 バイト未満) は、通常はまったく圧縮されません。パケットの圧縮には CPU 時間が必要なので、小さなパケットを圧縮しようとすると、実際にパフォーマンスが低下することがあります。

一般に、圧縮のスレッショルド値を小さくすると、非常に低速なネットワークではパフォーマンスが向上し、値を大きくすると CPU 使用率の減少によってパフォーマンスが向上する場合があります。ただし、圧縮のスレッショルド値を小さくするとクライアントとサーバの両方で CPU 使用率が増加するので、パフォーマンス分析を行って、圧縮のスレッショルドを変更するとパフォーマンスが向上するかどうかを判断してください。

「CompressionThreshold 接続パラメータ [COMPTH]」 238 ページと「-pt サーバ・オプション」 186 ページを参照してください。

♦ **SQL Anywhere 圧縮設定を調整するには、次の手順に従います。**

1. 通信の圧縮を有効にします。

   高度に圧縮可能なデータを大きなパケット・サイズで大量にデータ転送することで、最高の圧縮率を得ることができます。

   圧縮の有効化の詳細については、「Compress 接続パラメータ [COMP]」 237 ページと「-pc サーバ・オプション」 185 ページを参照してください。

2. CompressionThreshold 設定を調整します。

   圧縮スレッショルドの値を小さくすると、非常に低速なネットワークではパフォーマンスが向上し、値を大きくすると CPU 使用率の減少によってパフォーマンスが向上する場合があります。

   CompressionThreshold (COMPTH) 接続パラメータの調整については、「CompressionThreshold 接続パラメータ [COMPTH]」 238 ページと「-pt サーバ・オプション」 186 ページを参照してください。

ネットワーク通信のトラブルシューティング

ネットワーク・ソフトウェアにはさまざまなコンポーネントが含まれているため、問題が発生しやすくなります。ここで、ネットワーク・トラブルシューティングのヒントをいくつか説明しますが、ネットワークのトラブルシューティングを支援する一番の情報源は、ネットワーク通信ソフトウェア・ベンダから提供されるネットワーク通信ソフトウェアの資料と Sybase 製品の保守契約を結んでいるサポート・センタです。

ロギングの使用

-z データベース・サーバ・オプションを指定すると、トラブルシューティング用に、診断通信メッセージが表示され、その他のメッセージがサーバ・メッセージ・ウィンドウに表示されます。これにより、接続に失敗したときの状況や接続試行に使用された接続パラメータを診断したり、使用された通信リンクを特定したりできます。

互換性のあるプロトコルを使用していることの確認

クライアントとデータベース・サーバが同じプロトコルを使用していることを確認してください。サーバの -x オプションを使用して、サーバが使用するプロトコルのリストを選択します。また、CommLinks (LINKS) 接続パラメータを使用して、クライアント・アプリケーションが使用するプロトコルのリストを選択します。

これらのオプションを使用して、各アプリケーションが同じプロトコルを使用していることを確認できます。

デフォルトでは、ネットワーク・データベース・サーバは、使用可能なプロトコルをすべて使用します。サーバはどのアクティブ・プロトコルに対してもクライアント要求をサポートします。デフォルトでは、クライアントは共有メモリ・プロトコルのみを使用します。クライアントは CommLinks (LINKS) 接続パラメータを all に設定することにより、使用可能なすべてのプロトコルを使用できます。

参照

- 「-x サーバ・オプション」 205 ページ
- 「CommLinks 接続パラメータ [LINKS]」 235 ページ

最新のドライバがあることの確認

古いネットワーク・アダプタ・ドライバが原因で、通信上の問題が起こる可能性があります。使用しているネットワーク・アダプタが最新バージョンであることを確認してください。最新のネットワーク・アダプタ・ドライバは、ネットワーク・カードの製造業者または販売元から入手できます。

TCP/IP プロトコルのテスト

TCP/IP がインストールされ、正しく設定されていることをテストするには、ping ユーティリティが役に立ちます。

ping を使用した IP レイヤのテスト

各 IP レイヤには、4 つの整数をドットで区切った数字 (191.72.109.12 など) を使って、IPv4 アドレスが関連付けられています。ping は引数に IP アドレスを取り、そのアドレスに 1 つのパケットを送信しようとします。

まず、現在使用中のコンピュータが正しく構成されていることを確認するため、ping ユーティリティを使用して現在使用中のコンピュータの検出を試します。IP アドレスが 191.72.109.12 の場合、コマンド・プロンプトで次のコマンドを入力し、パケットがルート指定されるかどうかを確認します。

```
ping 191.72.109.12
```

パケットがルート指定されていれば、次のような出力が表示されます。

```
c:> ping 191.72.109.12
Pinging 191.72.109.12 with 32 bytes of data:
Reply from 191.72.109.12: bytes=32 time<.10ms TTL=32
Reply from 191.72.109.12: bytes=32 time<.10ms TTL=32
Reply from 191.72.109.12: bytes=32 time<.10ms TTL=32
...
```

これは、このコンピュータがパケットをコンピュータ自身に送信できることを示します。これで、IP レイヤが正しく設定されていることを十分に確認できました。TCP/IP を実行している他のユーザに IP アドレスを尋ね、ping ユーティリティを使用してそのコンピュータを検出できるかどうかを試すこともできます。

クライアント・コンピュータからデータベース・サーバを実行しているコンピュータに ping できることを確認してから、先へ進んでください。

IPv6 ネットワーク上のホストに接続する場合は、まず IPv6 がクライアント・コンピュータにインストールされていることを確認する必要があります。Windows XP の場合は、コマンド ipv6 install を実行して IPv6 をインストールします。Windows Vista の場合は、デフォルトで IPv6 がインストールされます。UNIX の場合、IPv6 のインストールはオペレーティング・システムごとに異なるので、IPv6 を有効にする手順についてはオペレーティング・システムのマニュアルを参照してください。

IPv6 をインストールして有効にしたら、ping6 コマンドを使用して、上述の ping コマンドの場合と同じ手順を実行します。次に例を示します。

```
c:> ping6 fe80::213:ceff:fe24:ca6

Pinging fe80::213:ceff:fe24:ca6
from fe80::213:ceff:fe24:ca6%6 with 32 bytes of data:

Reply from fe80::213:ceff:fe24:ca6%6: bytes=32 time<1ms
Reply from fe80::213:ceff:fe24:ca6%6: bytes=32 time<1ms
Reply from fe80::213:ceff:fe24:ca6%6: bytes=32 time<1ms
...
```


配線問題の診断

ネットワーク配線またはコネクタが不完全な場合は、発見しにくい問題を引き起こす可能性があります。同じようなコンピュータ上で同じ設定を使って、問題を再現してみます。あるコンピュータでだけ問題が発生する場合は、配線の問題かハードウェアの問題です。

頻度が高い問題のチェックリスト

次のリストに、頻度が高い問題のいくつかとその解決策を示します。

データベースまたはデータベース・サーバへの接続に関するトラブルシューティングについては、「接続のトラブルシューティング」 88 ページと「サーバ起動時のトラブルシューティング」 54 ページを参照してください。

接続しようとしたときに「**データベース・サーバが見つかりません。**」というメッセージを受信した場合は、クライアントがネットワークでデータベース・サーバを検索できていません。次のような問題がないか調べてください。

- TCP/IP プロトコルで、クライアントは要求をブロードキャストすることによって、データベース・サーバを検索した。このようなブロードキャストは、通常はゲートウェイを通過しないため、別の (サブ) ネットワーク上のコンピュータにあるデータベース・サーバを検索できません。この場合は、HOST (IP) プロトコル・オプションを使用して、サーバを実行しているコンピュータのホスト名を指定してください。
- クライアントとサーバの間にファイアウォールがあり、接続が妨害されている可能性がある。「ファイアウォール経由の接続」 124 ページを参照してください。
- パーソナル・サーバは同じコンピュータからの接続だけを受け入れている。クライアントとサーバが異なるコンピュータ上にある場合は、ネットワーク・サーバを使用する必要があります。
- ネットワーク・ドライバが正しくインストールされていないか、またはネットワーク配線が正しくない。
- 「**要求された通信リンクの初期化を無効にします**」というメッセージを受信し、リンクの起動に失敗した。原因として、ネットワーク・ドライバがインストールされていないことが考えられます。ネットワークのマニュアルを参照して、使用するドライバのインストール方法を確認してください。
- jConnect 経由で接続している場合、サーバが TCP/IP プロトコルを使用する必要がある。
- ローカル・コンピュータ上のデータベースに接続する場合、[接続] ダイアログの [データベース] タブにある [ネットワーク上でデータベース・サーバを検索] オプションがクリアされているかを確認する。ローカル・コンピュータ以外のコンピュータで稼働しているデータベース・サーバに接続する場合にこのオプションを選択できます。

ネットワーク・プロトコル・オプションの詳細については、「ネットワーク・プロトコル・オプション」 265 ページを参照してください。

タイムアウト値の調整

接続が予期せずに終了してしまう場合は、活性タイムアウトまたはアイドル・タイムアウトの値の調整を検討してください。

参照

- 「LivenessTimeout 接続パラメータ [LTO]」 255 ページ
- 「-tl サーバ・オプション」 197 ページ
- 「Idle 接続パラメータ」 251 ページ
- 「-ti サーバ・オプション」 197 ページ

第 5 章

データベース・サーバ

目次

SQL Anywhere データベース・サーバ

機能

パーソナル・データベース・サーバまたはネットワーク・データベース・サーバを起動します。

構文

{ **dbeng10** | **dbsrv10** }
[*server-options*] [*database-file* [*database-options*] ⋯]

NetWare を使用している場合の構文

[**load**] **dbsrv10** [*server-options*] [*database-file* [*database-options*] ⋯]

サーバ・オプション

サーバ・オプション	説明
@*data*	設定ファイルまたは環境変数からオプションを読み込みます。「@data サーバ・オプション」 147 ページを参照してください。
-?	使用法を表示します。「-? サーバ・オプション」 148 ページを参照してください。
-b	バルク・オペレーション・モードで実行します。「-b サーバ・オプション」 148 ページを参照してください。
-c *size*	初期キャッシュ・サイズを設定します。「-c サーバ・オプション」 149 ページを参照してください。
-ca 0	動的なキャッシュ・サイズの設定を無効にします (Windows、UNIX)。「-ca サーバ・オプション」 151 ページを参照してください。
-cc { **+** | **-** }	キャッシュ・ウォーミングに使用するデータベース・ページに関する情報を収集します。「-cc サーバ・オプション」 152 ページを参照してください。
-ch *size*	キャッシュ・サイズの上限値を設定します (Windows、UNIX)。「-ch サーバ・オプション」 152 ページを参照してください。
-cl *size*	キャッシュ・サイズの下限値を設定します (Windows、UNIX)。「-cl サーバ・オプション」 153 ページを参照してください。
-cm *size*	Address Windowing Extensions (AWE) キャッシュに割り付けるアドレス領域のサイズを指定します (Windows)。「-cm サーバ・オプション」 154 ページを参照してください。
-cr { **+** | **-** }	データベース・ページを保持するキャッシュを準備します。「-cr サーバ・オプション」 155 ページを参照してください。
-cs	サーバ・メッセージ・ウィンドウにキャッシュの使用状況を表示します。「-cs サーバ・オプション」 156 ページを参照してください。

サーバ・オプション	説明
-cv { **+** | **-** }	データベース・サーバ・ウィンドウでのキャッシュ・ウォーミングに関するメッセージの表示を制御します。「-cv サーバ・オプション」 156 ページを参照してください。
-cw	データベース・サーバ・キャッシュ・サイズの設定を目的とした Windows での Address Windowing Extensions の使用を有効にします。「-cw サーバ・オプション」 157 ページを参照してください。
-dt	テンポラリ・ファイルを保存するディレクトリを指定します。「-dt サーバ・オプション」 160 ページを参照してください。
-ec *encryption-options*	パケットの暗号化を有効にします (ネットワーク・サーバ)。「-ec サーバ・オプション」 161 ページを参照してください。
-ep	暗号化キーを入力するよう要求します。「-ep サーバ・オプション」 164 ページを参照してください。
-fc	ファイル・システム・フルのコールバック関数を含む DLL のファイル名を指定します。「-fc サーバ・オプション」 165 ページを参照してください。
-fips	強力なデータベースおよび通信の暗号化において FIPS 承認のアルゴリズムの使用を必須とします。「-fips サーバ・オプション」 166 ページを参照してください。
-ga	最後の接続を閉じた後、データベースを自動的にアンロードします。さらに、最後のデータベースを閉じた後で停止します (NetWare を除く)。「-ga サーバ・オプション」 167 ページを参照してください。
-gb *level*	データベース・プロセスの優先度クラスを *level* に設定します (Windows)。「-gb サーバ・オプション」 167 ページを参照してください。
-gc *num*	最大チェックポイント・タイムアウト時間を *num* 分に設定します。「-gc サーバ・オプション」 168 ページを参照してください。
-gd *level*	データベース起動パーミッションを設定します。「-gd サーバ・オプション」 168 ページを参照してください。
-ge *size*	外部関数を実行するスレッドのスタック・サイズを設定します。「-ge サーバ・オプション」 169 ページを参照してください。
-gf	トリガの起動を無効にします。「-gf サーバ・オプション」 170 ページを参照してください。
-gk *level*	サーバを停止するためのパーミッションを設定します。「-gk サーバ・オプション」 170 ページを参照してください。
-gl *level*	データをロードまたはアンロードするためのパーミッションを設定します。「-gl サーバ・オプション」 171 ページを参照してください。
-gm *num*	接続の最大数を設定します。「-gm サーバ・オプション」 171 ページを参照してください。

サーバ・オプション	説明
-gn *num*	データベース・サーバが同時に実行できるタスクの最大数を設定します。「-gn サーバ・オプション」 172 ページを参照してください。
-gp *size*	最大ページ・サイズを *size* バイトに設定します。「-gp サーバ・オプション」 173 ページを参照してください。
-gr *minutes*	最大リカバリ時間を *num* 分に設定します。「-gr サーバ・オプション」 173 ページを参照してください。
-gss *size*	スレッドのスタック・サイズを *size* バイトに設定します (Windows を除く)。「-gss サーバ・オプション」 174 ページを参照してください。
-gt *num*	使用できる物理プロセッサの最大数を設定します (ライセンスされたプロセッサの数を上限とする)。このオプションは、マルチプロセッサ・システムでのみ役立ちます。「-gt サーバ・オプション」 174 ページを参照してください。
-gtc *logical-processors-to-use*	データベース・サーバが許容するプロセッサ同時実行性の最大値を指定します。「-gtc サーバ・オプション」 175 ページを参照してください。
-gu *level*	ユーティリティ・コマンドのパーミッション・レベルを utility_db、all、none、DBA のいずれかに設定します。「-gu サーバ・オプション」 176 ページを参照してください。
-k	パフォーマンス・モニタ統計値の収集を制御します。「-k サーバ・オプション」 177 ページを参照してください。
-kl *GSS-API-library-file*	Kerberos GSS-API ライブラリ (UNIX では共有オブジェクト) のファイル名を指定し、データベース・サーバへの Kerberos 認証接続を有効にします。「-kl サーバ・オプション」 178 ページを参照してください。
-kr *server-realm*	Kerberos サーバ・プリンシパルの領域を指定し、データベース・サーバへの Kerberos 認証接続を有効にします。「-kr サーバ・オプション」 178 ページを参照してください。
-krb	データベース・サーバへの Kerberos 認証接続を有効にします。「-krb サーバ・オプション」 179 ページを参照してください。
-m	すべてのデータベースについて、各チェックポイント以降のトランザクション・ログをトランケートします。「-m サーバ・オプション」 180 ページを参照してください。
-n *name*	データベース・サーバ名を *name* にします。-n オプションは、指定された位置によって意味が変わるので注意してください。「-n サーバ・オプション」 181 ページを参照してください。
-o *filename*	指定したファイルにメッセージを出力します。「-o サーバ・オプション」 182 ページを参照してください。

サーバ・オプション	説明
-oe *filename*	起動エラー、致命的なエラー、アサーションをログするファイルを指定します。「-oe サーバ・オプション」 183 ページを参照してください。
-on *size*	コンソール・ログの最大サイズを指定します。ログ・ファイルがこのサイズに達すると、現在のファイルが拡張子 *old* を持つ名前に変更され、新しいファイルが作成されます。「-on サーバ・オプション」 183 ページを参照してください。
-os *size*	メッセージ用のログ・ファイルのサイズを制限します。「-os サーバ・オプション」 184 ページを参照してください。
-ot *filename*	コンソール・ログをトランケートし、出力メッセージを追加します。「-ot サーバ・オプション」 185 ページを参照してください。
-p *packet-size*	最大ネットワーク・パケット・サイズを設定します (ネットワーク・サーバ)。「-p サーバ・オプション」 185 ページを参照してください。
-pc	同一コンピュータ接続以外のすべての接続のパケットを圧縮します。「-pc サーバ・オプション」 185 ページを参照してください。
-pt *size_in_bytes*	圧縮を適用する最小のネットワーク・パケット・サイズを設定します。「-pt サーバ・オプション」 186 ページを参照してください。
-qi	データベース・サーバ・トレイ・アイコンまたはサーバ・メッセージ・ウィンドウを非表示にします (Windows)。「-qi サーバ・オプション」 187 ページを参照してください。
-qn	起動時にサーバ・メッセージ・ウィンドウを最小化しません (Windows と Linux)。「-qn サーバ・オプション」 187 ページを参照してください。
-qp	サーバ・メッセージ・ウィンドウにパフォーマンスのメッセージを表示しません。「-qp サーバ・オプション」 188 ページを参照してください。
-qs	起動エラー・ダイアログを表示しないようにします。「-qs サーバ・オプション」 188 ページを参照してください。
-qw	データベース・サーバ画面を表示しないようにします。「-qw サーバ・オプション」 189 ページを参照してください。
-r	読み込み専用モードでデータベースを開きます。「-r サーバ・オプション」 189 ページを参照してください。
-s *facility-ID*	syslog facility ID を設定します (UNIX)。「-s サーバ・オプション」 190 ページを参照してください。
-sb { **0** | **1** }	ブロードキャストに対するサーバの動作を指定します。「-sb サーバ・オプション」 191 ページを参照してください。
-sf *feature-list*	このデータベース・サーバで実行されるデータベースの機能を保護します。「-sf サーバ・オプション」 192 ページを参照してください。

サーバ・オプション	説明
-sk *key*	データベース・サーバで無効になっている機能を有効にするためのキーを指定します。「-sk サーバ・オプション」 195 ページを参照してください。
-su *password*	ユーティリティ・データベース (utility_db) の DBA ユーザのパスワードを設定します。または、ユーティリティ・データベースへの接続を無効にします。「-su サーバ・オプション」 196 ページを参照してください。
-ti *minutes*	シャットダウンするまでのクライアントのアイドル時間を設定します (デフォルト値は 240 分)。「-ti サーバ・オプション」 197 ページを参照してください。
-tl *seconds*	デフォルトのクライアント活性タイムアウト (秒数) を設定します (デフォルト値は 120 秒)。「-tl サーバ・オプション」 197 ページを参照してください。
-tmf	トランザクション・マネージャに、強制的に分散トランザクションをリカバリさせます (Windows)。「-tmf サーバ・オプション」 198 ページを参照してください。
-tmt *milliseconds*	分散トランザクションの再エンリスト・タイムアウトを設定します (Windows)。「-tmt サーバ・オプション」 199 ページを参照してください。
-tq *time*	終了時刻を設定します (ネットワーク・サーバ)。「-tq サーバ・オプション」 199 ページを参照してください。
-u	バッファ・ディスク I/O を使用します (Windows、UNIX)。「-u サーバ・オプション」 200 ページを参照してください。
-ua	非同期 I/O の使用をオフにします (Linux)。「-ua サーバ・オプション」 200 ページを参照してください。
-uc	データベース・サーバをコンソール・モードで起動します (UNIX)。「-uc サーバ・オプション」 201 ページを参照してください。
-ud	デーモンとして実行します (UNIX)。「-ud サーバ・オプション」 201 ページを参照してください。
-uf	致命的なエラーの発生時に実行するアクションを指定します (UNIX)。「-uf サーバ・オプション」 202 ページを参照してください。
-ui	[サーバ起動オプション] ダイアログを開いてサーバ・メッセージ・ウィンドウを表示するか、使用可能な表示がない場合はコンソール・モードでデータベース・サーバを起動します (Linux)。「-ui サーバ・オプション」 202 ページを参照してください。
-ut *minutes*	*min* 分ごとにテンポラリ・ファイルにタッチします (UNIX)。「-ut サーバ・オプション」 203 ページを参照してください。

サーバ・オプション	説明
-ux	サーバ・メッセージ・ウィンドウと [サーバ起動オプション] ダイアログを表示します (Linux)。「-ux サーバ・オプション」 203 ページを参照してください。
-v	データベース・サーバのバージョンを表示して停止します。「-v サーバ・オプション」 204 ページを参照してください。
-x *list*	カンマで区切られた通信リンクのリストを提供します。「-x サーバ・オプション」 205 ページを参照してください。
-xa *authentication-info*	監視サーバに対するデータベース名と認証文字列のリストを指定します。「-xa サーバ・オプション」 207 ページを参照してください。
-xf *state-file*	データベース・ミラーリング・システムに関するステータス情報の管理に使用されるファイルのロケーションを指定します。「-xf サーバ・オプション」 207 ページを参照してください。
-xs	サーバ側の Web サービス通信プロトコルを指定します。「-xs サーバ・オプション」 208 ページを参照してください。
-z	通信リンクに関する診断情報を提供します (ネットワーク・サーバ)。「-z サーバ・オプション」 210 ページを参照してください。
-ze	データベース・サーバ環境変数をサーバ・メッセージ・ウィンドウに表示します。「-ze オプション」 210 ページを参照してください。
-zl	各接続の最後に作成された SQL 文の取得をオンにします。「-zl サーバ・オプション」 211 ページを参照してください。
-zn	保管する要求ログ・ファイルのコピー数を指定します。「-zn サーバ・オプション」 211 ページを参照してください。
-zo *filename*	要求ロギング情報を別個のファイルにリダイレクトします。「-zo サーバ・オプション」 212 ページを参照してください。
-zp	クエリ・オプティマイザが最近使用したプランの取得をオンにします。「-zp サーバ・オプション」 213 ページを参照してください。
-zr { **all** | **SQL** | **none** }	SQL オペレーションのログを有効にします。デフォルトは、none です。「-zr サーバ・オプション」 213 ページを参照してください。
-zs *size*	要求ロギング用のログ・ファイルのサイズを制限します。「-zs サーバ・オプション」 215 ページを参照してください。
-zt	要求タイミング情報のロギングをオンにします。「-zt オプション」 216 ページを参照してください。

リカバリ・オプション

リカバリ・オプション	説明
-f	トランザクション・ログなしでデータベースを強制的に起動します。「-f リカバリ・オプション」 217 ページを参照してください。

データベース・オプション

次のオプションは、*必ず*データベース・サーバ・コマンドでデータベース・ファイル名の後に指定してください。

データベース・オプション	説明
-a *filename*	名前付きトランザクション・ログ・ファイルを適用します。「-a データベース・オプション」 218 ページを参照してください。
-ad *log-directory*	データベースに適用されるログ・ファイルがあるディレクトリを指定します。「-ad データベース・オプション」 218 ページを参照してください。
-ar	トランザクション・ログと同じディレクトリ内にあるログ・ファイルをデータベースに適用します。「-ar データベース・オプション」 219 ページを参照してください。
-as	トランザクション・ログの適用後もデータベースの実行を継続します (-ad または -ar とともに使用)。「-as データベース・オプション」 220 ページを参照してください。
-ds	データベースのすべての DB 領域のロケーションを指定します。「-ds データベース・オプション」 221 ページを参照してください。
-dh	このサーバに対して dblocate が実行された場合、データベースを表示しません。「-dh データベース・オプション」 222 ページを参照してください。
-ek *key*	暗号化キーを指定します。「-ek データベース・オプション」 222 ページを参照してください。
-m	指定のデータベースについて、チェックポイント以降のトランザクション・ログをトランケート (削除) します。「-m データベース・オプション」 223 ページを参照してください。
-n *name*	データベースに名前を付けます。「-n データベース・オプション」 223 ページを参照してください。
-r	読み込み専用モードで指定のデータベースを開きます。データベースは変更できません。「-r データベース・オプション」 224 ページを参照してください。
-sn *alternate-server-name*	データベース・サーバ上で動作する 1 つのデータベースに代替サーバ名を割り当てます。「-sn オプション」 225 ページを参照してください。

データベース・オプション	説明
-xp *mirroring-options*	データベース・ミラーリングが使用されている場合に、稼働しているサーバにパートナーと監視サーバへの接続を可能にする情報を提供します。「-xp データベース・オプション」 227 ページを参照してください。

備考

dbeng10 コマンドは、パーソナル・データベース・サーバを起動します。**dbsrv10** コマンドは、ネットワーク・データベース・サーバを起動します。

キャッシュ・サイズ

データベース・サーバに割り当て可能なキャッシュ・メモリの容量は、パフォーマンスに影響を及ぼす主要な要因の 1 つになります。データベース・サーバは、-c オプションで指定された値またはデフォルト値をキャッシュ・メモリの初期容量として使用します。

デフォルトのキャッシュ・サイズの詳細については、「-c サーバ・オプション」 149 ページを参照してください。

Windows と UNIX では、データベース・サーバはヒューリスティック・アルゴリズムに基づき、使用するメモリを必要に応じて自動的に増加させます。

「パフォーマンス向上へのキャッシュの使用」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

データベース・オプションを使用して、キャッシュの上限値を設定できます。「-ch サーバ・オプション」 152 ページを参照してください。

また、キャッシュ容量を初期容量のままにしておくこともできます。「-ca サーバ・オプション」 151 ページを参照してください。

サーバ間の相違点

パーソナル・データベース・サーバでは最大 10 の同時接続を使用でき、要求処理には最大 1 つの CPU を使用し、ネットワーク・クライアント／サーバ接続はサポートしません。

さらに、新規データベースを開始するために必要なデフォルトのパーミッション・レベルや、CHECKPOINT 文を実行するために必要なパーミッションなどの小さな相違点もあります。

プラットフォームの可用性

サポートされている各オペレーティング・システムで、パーソナル・データベース・サーバとネットワーク・データベース・サーバの両方が提供されていますが、次の例外があります。

- **Novell NetWare**　ネットワーク・サーバだけが提供されています。
- **Windows CE**　ネットワーク・サーバだけが提供されています。ネットワーク・サーバでは TCP/IP がサポートされるため、使用しているデスクトップ・コンピュータから Sybase Central によってデータベース管理などのタスクを実行できます。

NetWare に関する注意

NetWare では、SQL Anywhere の各メジャー・バージョンにつき 1 つのデータベース・サーバのみ実行できます。同じメジャー・バージョンを使用してデータベース・サーバをもう 1 つ実行しようとしても、エクスポートされた関数と共有メモリが SPX 上で競合するため、サーバの起動は失敗します。

NetWare では、データベース・ファイルとトランザクション・ログ・ファイルを、NetWare ボリューム上に置き、フル・パスを指定してください。NetWare では、2 つ以上のハード・ディスクにまたがるボリュームを使用できます。

データベース・ファイル　*database-file* には、データベース・ファイル名を指定します。*database-file* にファイル名を拡張子なしで指定すると、SQL Anywhere は *database-file* に拡張子 *.db* を付けてファイルを検索します。

相対パスを使用する場合、そのパスは現在の作業ディレクトリと相対関係となります。フル・パスも指定できます。

Windows イベント・ログ・メッセージを出力しない

レジストリ・エントリを設定することによって、Windows イベント・ログ・エントリの出力を抑制できます。レジストリ・エントリは、*Software¥Sybase¥SQL Anywhere¥10.0* です。

イベント・ログ・エントリを制御するには、EventLogMask キーを設定します。このキーは REG_DWORD タイプを指定するものです。この値はビット・マスクで、さまざまなイベント・メッセージに関する内部ビット値を保有します。

```
errors      EVENTLOG_ERROR_TYPE       0x0001
warnings    EVENTLOG_WARNING_TYPE     0x0002
information EVENTLOG_INFORMATION_TYPE 0x0004
```

たとえば、EventLogMask キーを 0 に設定すると、メッセージはまったく出力されなくなります。推奨される設定は 1 です。情報メッセージと警告メッセージは出力されませんが、エラー・メッセージは出力されます。デフォルト設定 (エントリが存在しない場合) では、すべてのメッセージ・タイプが出力されます。

参照

- 「データベース・サーバの実行」 11 ページ
- 「ネットワーク・プロトコル・オプション」 265 ページ

クワイエット・モードで作動する

データベース・サーバはクワイエット・モードをサポートしています。サーバをどのようなクワイエット・モードで操作するかを決定します。クワイエット・モードの範囲には、メッセージやシステム・トレイのアイコンを非表示にするモードから、完全に非表示にするモードまであります。Windows 上のデータベース・サーバを完全に非表示にするモードで操作するには、-qi および -qs オプションを指定します。これらのオプションを設定すると、すべてのアイコンとすべての起動エラー・メッセージが表示されなくなるので、サーバが実行中であることを視覚的に表示するものがなくなります。データベース・サーバをクワイエット・モードで実行する場合、-o または -oe オプションのいずれか (または両方) を使用してエラーを診断できます。

-qi と -qs オプションを使用しても、-v (バージョン) と -ep (データベース暗号化パスワードのプロンプト) サーバ・オプションによるダイアログは表示されることに注意してください。

データベース・サーバ・オプション

次のオプションは、個々のデータベースだけでなく、サーバ全体に適用されます。

@data サーバ・オプション

指定された環境変数または設定ファイルからオプションを読み込みます。

構文

{ **dbsrv10** | **dbeng10** } **@***data* …

適用対象

すべてのオペレーティング・システムとサーバのほか、言語選択ユーティリティ (dblang)、再構築ユーティリティ (rebuild)、証明書の作成ユーティリティ (createcert)、証明書ビューワ・ユーティリティ (viewcert)、ActiveSync プロバイダ・インストール・ユーティリティ (mlasinst)、およびファイル非表示ユーティリティ (dbfhide) を除くすべてのデータベース・ユーティリティ

備考

このオプションを使用して、指定された環境変数または設定ファイルからコマンドライン・オプションを読み出します。両方が指定された名前と同じ場合は、変数値が使用されます。

設定ファイルには、改行を含めたり、あらゆるオプションの設定を格納したりできます。「設定ファイルの使用」 637 ページを参照してください。

設定ファイルの情報を保護する (たとえばパスワードが含まれるため) 場合は、ファイル非表示 (dbfhide) ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。

@data パラメータはコマンド・ラインの任意の位置に指定でき、ファイルに含まれるパラメータがその位置に挿入されます。複数のファイルを指定可能で、ファイル指定子をコマンド・ライン・オプションで使用できます。

例

次の設定ファイルには、**myserver** という名前のサーバをキャッシュ・サイズ 4 MB で起動し、サンプル・データベースをロードする 1 組のオプションが含まれています。

```
-c 4096
-n myserver
"c:¥mydatabase.db"
```

この設定ファイルを *c:¥config.txt* として保存すると、コマンドで次のように使用できます。

```
dbsrv10 @c:¥config.txt
```

次の設定ファイルにはコメントが含まれています。


```
#This is the server name:
-n MyServer
#These are the protocols:
-x tcpip
#This is the database file
my.db
```

次の文 (すべて 1 行で入力) は、データベース・サーバをキャッシュ・サイズ 4 MB で起動し、サンプル・データベースをロードするオプションを格納する環境変数を設定します。

```
set envvar=-c 4096 "c:¥mydatabase.db"
```

次の文は、**envvar** という環境変数を使用してデータベース・サーバを起動します。

```
dbsrv10 @envvar
```

-? サーバ・オプション

使用法を表示します。

構文

{ **dbsrv10** | **dbeng10** } **-?**

適用対象

すべてのオペレーティング・システムとサーバ

備考

各サーバ・オプションの簡単な説明を表示します。データベースは、それ以外のタスクは実行しません。

-b サーバ・オプション

バルク・オペレーション・モードを使用します。

構文

{ **dbsrv10** | **dbeng10** } **-b** …

適用対象

すべてのオペレーティング・システムとサーバ

備考

これは、Interactive SQL INPUT コマンドを使用して、大量のデータをデータベースにロードする場合に役立ちます。

LOAD TABLE を使用してデータをバルク・ロードする場合は、-b オプションは使用しないでください。

このオプションを使用した場合、データベース・サーバでは、1 つのアプリケーションで 1 つの接続しか確立できなくなります。ロールバック・ログは保存されますが、トランザクション・ログは保存されません。マルチユーザ・ロッキング・メカニズムはオフになっています。

-b オプションでデータをロードした後、初めてデータベース・サーバを起動する場合は、新しいログ・ファイルを使用してください。

バルク・オペレーション・モードにしても、トリガは起動不可になりません。

-c サーバ・オプション

データベース・ページや他のサーバ情報をキャッシュするために予約される初期メモリを設定します。

構文

{ **dbsrv10** | **dbeng10** } **-c** { *size*[**k** | **m** | **g** | **p**] } …

適用対象

すべてのオペレーティング・システムとサーバ

備考

データベース・サーバ・キャッシュで使用できるキャッシュ・メモリの量は、パフォーマンスを制御する主要な要因の 1 つになります。初期のキャッシュ・メモリ量は、-c サーバ・オプションを使用して設定できます。サーバ用のキャッシュ・メモリが大きければ大きいほど、パフォーマンスは向上します。

size には、メモリ容量をバイト単位で指定します。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ **k**、**m**、または **g** を使用してください。

単位 **p** は、物理システム・メモリと空きアドレス領域のうち、いずれか小さい方のパーセンテージを表します。空きアドレス領域のサイズはオペレーティング・システムによって異なります。次に例を示します。

- 2.8 GB − Windows 32 ビット Advanced Server、Enterprise Server、Datacenter Server、Vista
- 3.8 GB − Windows x64 Edition 上の 32 ビット・データベース・サーバ
- 1.8 GB − その他すべての 32 ビット・システム
- 利用可能なアドレス領域を利用可能な物理システム・メモリのパーセンテージで表示 − Windows CE

p を使用すると、引数はパーセンテージを表します。p の代わりに % も使用できますが、UNIX 以外のオペレーティング・システムではほとんどの場合 % を環境変数のエスケープ文字として使用するため、% 文字をエスケープする必要があります。最小キャッシュ・サイズを物理システム・メモリの 50% に設定するには、次のコマンドを使用します。

```
dbeng10 -c 50%% ...
```

NetWare と UNIX オペレーティング・システムでは、キャッシュ・サイズは次のうちいずれか小さい方に設定されます。

- -c の後に指定された値

- (*メモリ・サイズ* - 5 MB) の 95 %

Windows CE では、キャッシュ・サイズは次のうちいずれか小さい方に設定されます。

- -c の後に指定された値
- (*メモリ・サイズ* - 2 MB) の 95 %

-c オプションを指定しないと、初期キャッシュ・メモリの割り当てサイズが次のように計算されます。

1. 各オペレーティング・システムの、デフォルトのキャッシュ・サイズを使用します。
   - **Windows CE** 600 KB
   - **Windows** 2 MB
   - **NetWare** 8 MB
   - **UNIX** 8 MB
2. ランタイムの最小デフォルト・キャッシュ・サイズを計算します。キャッシュ・サイズは、次のうち小さい方の数値になります。
   - コンピュータの物理メモリの 25 %
   - コマンド・ラインで指定されたメイン・データベース・ファイルの合計サイズ。メイン・データベース・ファイル以外の追加の DB 領域は、この計算の対象とはなりません。ファイルを指定しないと、この値は 0 になります。
3. 計算された 2 つの値のうち、大きい方のサイズが割り当てられます。

NetWare データベース・サーバ
データベース・サーバのメモリと NetWare ファイル・システム・バッファのメモリとの間にはトレードオフがあります。データベース・サーバ・キャッシュが大きくなると、NetWare ファイル・システムのパフォーマンスは低下しますが、データベース・サーバのパフォーマンスが向上します。データベース・サーバ・キャッシュが大きすぎると、キャッシュ・バッファのメモリが不十分であるというエラーがレポートされます。
ファイル・サーバに新しいディレクトリやファイルが追加されるたびに、NetWare が要求するメモリ領域は増加します。NetWare サーバ上でのメモリ使用状況を追跡するには、*monitor.nlm* をロードし (まだロードされていない場合)、[リソースの利用] を選択します。NetWare サーバ・コンピュータ用にメモリを追加すると、データベースのパフォーマンスかファイル・サーバのパフォーマンス、またはその両方が大幅に改善されます。

データベースを暗号化している場合は、キャッシュ・サイズを大きくすることもできます。また、動的なキャッシュ・サイズの変更 (-ca) を使用している場合は、使用されるキャッシュ・サイズが使用可能なメモリのサイズによって制限されることがあります。

「キャッシュ・サイズの拡大」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

サーバ・メッセージ・ウィンドウには起動時のキャッシュ・サイズが表示されます。また、次の文を使用して現在のキャッシュ・サイズを取得することもできます。

```
SELECT PROPERTY( 'CacheSize' )
```

参照

- 「-ca サーバ・オプション」 151 ページ
- 「-ch サーバ・オプション」 152 ページ
- 「-cl サーバ・オプション」 153 ページ
- 「キャッシュが使用するメモリの制限」 『SQL Anywhere サーバ - SQL の使用法』

例

次の例 (すべて 1 行で入力) は、**myserver** という名前のサーバをキャッシュ・サイズ 3 MB で起動し、サンプル・データベースをロードします。

```
dbeng10 -c 3m -n myserver "samples-dir¥demo.db"
```

samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

-ca サーバ・オプション

静的キャッシュ・サイズを強制的に適用します。

構文

{ dbsrv10 | dbeng10 } -ca 0 …

適用対象

Windows、UNIX

備考

サーバの負荷が高い場合は、コマンド・ラインで -ca 0 を指定して、自動キャッシュ増加機能を無効にできます。-ca 0 オプションの設定がない場合、データベース・サーバは自動的にキャッシュ・サイズを増加します。ただし、キャッシュを追加しないとデータベース・サーバで「**致命的エラー： 動的メモリが足りません。**」というエラーが発生するような場合、-ca 0 を使用してもキャッシュ・サイズが増加します。

このサーバ・オプションは、-ca 0 という形式でのみ使用してください。

参照

- 「-c サーバ・オプション」 149 ページ
- 「-ch サーバ・オプション」 152 ページ
- 「-cl サーバ・オプション」 153 ページ
- 「キャッシュが使用するメモリの制限」 『SQL Anywhere サーバ - SQL の使用法』

例

次の例は、**myserver** という名前のサーバを、使用可能な物理メモリの 40% の静的キャッシュを保持して起動し、サンプル・データベースをロードします。


```
dbsrv10 -c 40P -ca 0 -n myserver "samples-dir¥demo.db"
```

samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

-cc サーバ・オプション

次回にデータベースが起動されるときに、キャッシュ・ウォーミングに使用するデータベース・ページに関する情報を収集します。

構文

{ dbsrv10 | dbeng10 } -cc { + | - } ...

適用対象

すべてのオペレーティング・システムとサーバ

備考

デフォルトでは、ページ収集はオンになっています。収集がオンになると、データベース・サーバは要求された各データベース・ページを追跡し続けます。ページの収集は、最大ページ数が収集されるか、データベースが停止されるか、または収集率が最小値を下回った場合に終了します。収集する最大ページ数を設定したり、収集率の値を指定したりすることはできません (この値は、キャッシュ・サイズとデータベース・サイズに基づいています)。いったん収集が停止すると、要求されたページに関する情報はデータベースに記録されるので、それらのページは次回データベースが -cr オプションで開始されるときに、キャッシュの準備に使用できます。参照されたページの収集は、デフォルトではオンになっています。

参照

- 「-cr サーバ・オプション」 155 ページ
- 「-cv サーバ・オプション」 156 ページ
- 「キャッシュ・ウォーミングの使用」 『SQL Anywhere サーバ - SQL の使用法』

-ch サーバ・オプション

自動キャッシュ増加機能を利用した際の最大キャッシュ・サイズ上限値を設定します。

構文

{ dbsrv10 | dbeng10 } -ch { *size*[**k** | **m** | **g** | **p**] } ⋯

適用対象

Windows、UNIX

備考

このオプションは、データベース・サーバが自動キャッシュ増加機能を実行するときに使用できる、キャッシュ・サイズの上限を設定します。デフォルトでは、上限値の概算は、利用可能なアドレス領域とコンピュータの物理メモリの 90 % のうち、いずれか低い方になります。

size には、メモリ容量をバイト単位で指定します。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ **k**、**m**、または **g** を使用してください。

単位 **p** は、物理システム・メモリと空きアドレス領域のうち、いずれか小さい方のパーセンテージを表します。空きアドレス領域のサイズはオペレーティング・システムによって異なります。次に例を示します。

- 2.8 GB − Windows 32 ビット Advanced Server、Enterprise Server、Datacenter Server
- 3.8 GB − Windows x64 Edition 上の 32 ビット・データベース・サーバ
- 1.8 GB − その他すべての 32 ビット・システム
- 利用可能なアドレス領域を利用可能な物理システム・メモリのパーセンテージで表示 − Windows CE

p を使用すると、引数はパーセンテージを表します。P の代わりに % も使用できますが、UNIX 以外のオペレーティング・システムではほとんどの場合 % を環境変数のエスケープ文字として使用するため、% 文字をエスケープする必要があります。最小キャッシュ・サイズを物理システム・メモリの 50 % に設定するには、次のコマンドを使用します。

```
dbeng10 -ch 50%% ...
```

参照

- 「-c サーバ・オプション」 149 ページ
- 「-ca サーバ・オプション」 151 ページ
- 「-cl サーバ・オプション」 153 ページ
- 「キャッシュが使用するメモリの制限」 『SQL Anywhere サーバ - SQL の使用法』

例

次の例は、silver という名前のサーバを最大キャッシュ・サイズ 2 MB で起動し、サンプル・データベースをロードします。

```
dbeng10 -ch 2m -n silver "samples-dir¥demo.db"
```

samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

-cl サーバ・オプション

自動キャッシュ・サイズ変更機能に対して最小キャッシュ・サイズを設定します。

構文

{ **dbsrv10** | **dbeng10** } **-cl** { *size*[**k** | **m** | **g** | **p**] } …

適用対象

Windows、UNIX

備考

このオプションは、キャッシュの下限値を設定します。-c オプションを使用して初期キャッシュ・サイズを指定すると、最小キャッシュ・サイズは初期キャッシュ・サイズと同じになります。初期キャッシュ・サイズを指定しない場合、デフォルトの初期キャッシュ・サイズは Windows では 2 MB、UNIX では 8 MB です。

size には、メモリ容量をバイト単位で指定します。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ **k**、**m**、または **g** を使用してください。

単位 **p** は、物理システム・メモリと空きアドレス領域のうち、いずれか小さい方のパーセンテージを表します。空きアドレス領域のサイズはオペレーティング・システムによって異なります。次に例を示します。

- 2.8 GB − Windows 32 ビット Advanced Server、Enterprise Server、Datacenter Server
- 3.8 GB − Windows x64 Edition 上の 32 ビット・データベース・サーバ
- 1.8 GB − その他すべての 32 ビット・システム
- 利用可能なアドレス領域を利用可能な物理システム・メモリのパーセンテージで表示 − Windows CE

p を使用すると、引数はパーセンテージを表します。P の代わりに % も使用できますが、UNIX 以外のオペレーティング・システムではほとんどの場合 % を環境変数のエスケープ文字として使用するため、% 文字をエスケープする必要があります。最小キャッシュ・サイズを物理システム・メモリの 50% に設定するには、次のコマンドを使用します。

```
dbeng10 -cl 50%% ...
```

参照

- 「-c サーバ・オプション」 149 ページ
- 「-ca サーバ・オプション」 151 ページ
- 「-ch サーバ・オプション」 152 ページ
- 「キャッシュが使用するメモリの制限」 『SQL Anywhere サーバ - SQL の使用法』

例

次の例は、silver という名前のサーバを最小キャッシュ・サイズ 5 MB で起動し、データベース・ファイル *example.db* をロードします。

```
dbeng10 -cl 5m -n silver "c:¥example.db"
```

-cm サーバ・オプション

Windows において、Address Windowing Extensions (AWE) キャッシュに割り付けるアドレス領域のサイズを指定します。

構文

{ **dbsrv10** | **dbeng10** } **-cm** { *size*[**k** | **m** | **g** | **p**] } …

適用対象

Windows

備考

サポートされるプラットフォーム上で AWE キャッシュを使用する場合、データベース・サーバは 512 MB を除くアドレス領域全体を使用してキャッシュ・メモリにアクセスします。アドレス領域のうち 512 MB は、サーバがロードする必要のある DLL やキャッシュ以外のメモリ割り付けなどの目的に使用されます。ほとんどのシステムでは、このデフォルト設定でキャッシュ・サイズは十分です。確保されるアドレス領域のサイズを拡大または縮小する必要がある場合は、-cm オプションを指定します。データベース・サーバが使用しているアドレス領域のサイズは、起動時にサーバ・メッセージ・ウィンドウに表示されます。

size には、メモリ容量をバイト単位で指定します。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ **k**、**m**、または **g** を使用してください。

単位 **p** は空きアドレス領域のパーセンテージを表します。**p** を使用すると、引数はパーセンテージを表します。P の代わりに % も使用できますが、UNIX 以外のオペレーティング・システムではほとんどの場合 % を環境変数のエスケープ文字として使用するため、% 文字をエスケープする必要があります。最小キャッシュ・サイズをアドレス領域の 50 % に設定するには、次のコマンドを使用します。

```
dbeng10 -cm 50%% ...
```

参照

- 「-ca サーバ・オプション」 151 ページ
- 「-ch サーバ・オプション」 152 ページ
- 「-cl サーバ・オプション」 153 ページ
- 「-cw サーバ・オプション」 157 ページ
- 「キャッシュが使用するメモリの制限」 『SQL Anywhere サーバ - SQL の使用法』

-cr サーバ・オプション

データベースが最後に実行されたときに収集した情報を使用して、キャッシュとデータベース・ページを再ロード (準備) します。

構文

{ dbsrv10 | dbeng10 } -cr { + | - } ...

適用対象

すべてのオペレーティング・システムとサーバ

備考

データベースが最後に起動したときに参照されたページを使用して、データベース・サーバにキャッシュを準備するよう指示できます (ページ収集は -cc オプションを使用してオンにします)。キャッシュ・ウォーミングは、デフォルトではオンになっています。データベースが起動されると、サーバはデータベースをチェックして、データベースが最後に起動されたときに要求

されたページの収集があるかどうかを確認します。データベースにこの情報が含まれている場合、前に参照したページがキャッシュにロードされます。

データベースが最後に起動されたときに参照されたページのキャッシュを準備することで、同じクエリまたは似たようなクエリがデータベースの起動のたびに実行されるような場合に、パフォーマンスを改善できます。

参照

- 「-cc サーバ・オプション」 152 ページ
- 「-cv サーバ・オプション」 156 ページ
- 「キャッシュ・ウォーミングの使用」 『SQL Anywhere サーバ - SQL の使用法』

-cs サーバ・オプション

サーバ・メッセージ・ウィンドウにキャッシュ・サイズの変更情報を表示します。

構文

{ dbsrv10 | dbeng10 } -cs ⋯

適用対象

Windows、UNIX

備考

トラブルシューティングを目的とする場合、キャッシュ・サイズが変更されるたびに、サーバ・メッセージ・ウィンドウにキャッシュ情報を表示します。

-cv サーバ・オプション

データベース・サーバ・ウィンドウでのキャッシュ・ウォーミングに関するメッセージの表示を制御します。

構文

{ dbsrv10 | dbeng10 } -cv { + | - } ...

適用対象

すべてのオペレーティング・システムとサーバ

備考

-cv+ が指定されると、次のキャッシュ・ウォーミング・アクティビティが発生した場合に、サーバ・メッセージ・ウィンドウにメッセージが表示されます。

- 要求されたページの収集が開始または停止する (-cc サーバ・オプションで制御)
- ページの再ロードが開始または停止する (-cr サーバ・オプションで制御)

デフォルトでは、このオプションはオフです。

参照

- 「-cc サーバ・オプション」 152 ページ
- 「-cr サーバ・オプション」 155 ページ
- 「キャッシュ・ウォーミングの使用」 『SQL Anywhere サーバ - SQL の使用法』

例

次のコマンドは、データベース・ページの収集とページのロードをオンにして *mydatabase.db* というデータベースを起動し、これらのアクティビティに関するメッセージをサーバ・メッセージ・ウィンドウに表示します。

```
dbsrv10 -cc+ -cr+ -cv+ mydatabase.db
```

-cw サーバ・オプション

データベース・サーバ・キャッシュ・サイズの設定を目的とした Windows での Address Windowing Extensions (AWE) の使用を有効にします。

構文

{ dbsrv10 | dbeng10 } -cw …

適用対象

Windows

備考

データベース・サーバ・キャッシュで使用できるキャッシュ・メモリの量は、パフォーマンスを制御する主要な要因の 1 つになります。Windows は Address Windowing Extensions をサポートしているため、-cw オプションを使用して、システムに搭載されている最大物理メモリを基にサイズの大きいキャッシュを利用できます。

AWE キャッシュは、64 ビットの SQL Anywhere データベース・サーバではサポートされていません。

オペレーティング・システム	最大キャッシュ・サイズ (非 AWE)	Windows がサポートする最大メモリ・サイズ
Windows 2000 Professional	1.8 GB	4 GB
Windows 2000 Server	1.8 GB	4 GB
Windows 2000 Advanced Server	2.7 GB[1]	8 GB
Windows 2000 Datacenter Server	2.7 GB[1]	64 GB
Windows XP Home Edition	1.8 GB	2 GB
Windows XP Professional	1.8 GB	4 GB
Windows Server 2003、Web Edition	1.8 GB	2 GB

オペレーティング・システム	最大キャッシュ・サイズ (非 AWE)	Windows がサポートする最大メモリ・サイズ
Windows Server 2003、Standard Edition	1.8 GB	4 GB
Windows Server 2003、Enterprise Edition	2.7 GB[1]	32 GB
Windows Server 2003、Datacenter Edition	2.7 GB[1]	64 GB
Windows Vista Ultimate	2.7 GB[2]	4 GB
Windows Vista Enterprise	2.7 GB[2]	4 GB
Windows Vista Business	2.7 GB[2]	4 GB
Windows Vista Home Premium	2.7 GB[2]	4 GB
Windows Vista Home Basic	2.7 GB[2]	4 GB
Windows Vista Starter	2.7 GB[2]	1 GB

[1] Windows XP/200x の場合は、このサイズのキャッシュを使用するには、/3GB オプションを使用してオペレーティング・システムを起動する必要があります。

[2] Windows Vista の場合は、このサイズのキャッシュを使用するには、管理者として次のコマンドを実行した後にオペレーティング・システムを起動する必要があります。

```
bcdedit /set increaseuserva 3072
```

AWE キャッシュを使用している場合、システムで使用可能なほとんどすべての物理メモリをキャッシュに割り当てることができます。

AWE キャッシュ以外を使用して希望のサイズのキャッシュを設定できる場合は、そちらをおすすめします。AWE キャッシュで割り当てられるメモリはデータベース・サーバでしか使用できないためです。つまり、データベース・サーバの実行中は、オペレーティング・システムやその他のアプリケーションは、データベース・サーバ・キャッシュに割り当てられたメモリを使用できません。AWE キャッシュでは、動的なキャッシュ・サイズの変更はサポートされていません。このため、AWE キャッシュを使用するときに、-ch または -cl オプションを指定してキャッシュ・サイズの上限と下限を設定しても無視されます。

デフォルトでは、アドレス領域のうち 512 MB が SQL Anywhere AWE キャッシュ以外の目的に確保されます (アドレス領域とは、ある時点にプログラムがアクセスできるメモリの量をいいます)。ほとんどの場合は、この量で十分ですが、-cm オプションを使用して、確保されるアドレス領域のサイズを変更することもできます。「-cm サーバ・オプション」 154 ページを参照してください。

AWE キャッシュを使用してデータベース・サーバを起動するには、次の条件があります。

- Windows Vista の場合は、管理者としてデータベース・サーバを実行する必要があります。

- システムで使用可能なメモリが少なくとも 130 MB 必要です。
- Windows XP/200x では、システムのメモリが 2 GB ー 16 GB の場合、*boot.ini* ファイルの [operating systems] セクションの Windows のブート行に /3GB オプションを追加してください。

  Windows Vista では、システムのメモリが 2 GB ー 16 GB の場合、管理者として次のコマンドを実行する必要があります。

  ```
  bcdedit /set increaseuserva 3072e
  ```

  Windows XP/200x では、システムのメモリが 16 GB を超える場合は、*boot.ini* ファイルの [operating systems] セクションの Windows ブート行に /3GB オプションを追加しないでください。Windows では、16 GB を超えるメモリは処理できないからです。
- Windows XP/200x では、システムのメモリが 4 GB を超える場合は、*boot.ini* ファイルの [operating systems] セクションの Windows のブート行に /PAE オプションを追加してください。

  Windows Vista では、システムのメモリが 4 GB を超える場合は、管理者として次のコマンドを実行してください。

  ```
  bcdedit /set pae ForceEnable
  ```
- サーバを実行中のユーザ ID に「メモリにページをロック」権限を付与します。ここでは、Windows XP での手順を説明します。
  1. 管理者として Windows にログオンします。
  2. [コントロール パネル] を開きます。
  3. [管理ツール] をダブルクリックします。
  4. [ローカル セキュリティ ポリシー] をダブルクリックします。
  5. 左ウィンドウ枠の [ローカル ポリシー] を開きます。
  6. [ユーザー権利の割り当て] をダブルクリックします。
  7. 右ウィンドウ枠の [メモリ内のページのロック] ポリシーをダブルクリックします。
  8. [ユーザーまたはグループの追加] をクリックします。

     [ユーザーまたはグループの選択] ダイアログが表示されます。
  9. ユーザ名を入力し、[OK] をクリックします。
  10. [メモリ内のページのロック] プロパティ・シートで [OK] をクリックします。
  11. コンピュータを再起動して、設定を有効にします。

-cw オプションと -c オプションをコマンド・ラインで指定すると、データベース・サーバは次のように初期キャッシュを割り当てようとします。

1. AWE キャッシュは、-c オプションで指定されたキャッシュ・サイズよりも大きくなることはありません。-c オプションで指定された値が 2 MB より小さい場合、AWE は使用されません。
2. AWE キャッシュは、使用可能なすべての物理メモリから 128 MB を引いた容量を上回ることはありません。
3. AWE キャッシュは 2 MB より小さくなることはありません。この最小容量の物理メモリが使用可能でない場合、AWE キャッシュは使用されません。

-cw オプションを指定し、-c オプションを指定しないと、データベース・サーバは次のように初期キャッシュを割り当てようとします。

1. AWE キャッシュは、使用可能なメモリのうち、128 MB をオペレーティング・システムで使用して、それ以外の容量を 100 % 使用します。
2. AWE キャッシュは、コマンド・ラインで指定されたメイン・データベース・ファイルの合計サイズを上回ることはありません。メイン・データベース・ファイル以外の追加の DB 領域は、この計算の対象とはなりません。ファイルを指定しないと、この値は 0 になります。
3. AWE キャッシュは 2 MB より小さくなることはありません。この最小容量の物理メモリが使用可能でない場合、AWE キャッシュは使用されません。

サーバが AWE キャッシュを使用する場合、キャッシュ・ページ・サイズは 4 KB 以上となり、動的キャッシュ・サイズ決定は無効になります。

「パフォーマンス向上へのキャッシュの使用」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

参照

- 「-c サーバ・オプション」 149 ページ
- 「-ch サーバ・オプション」 152 ページ
- 「-cm サーバ・オプション」 154 ページ

例

次の例は、**myserver** という名前のサーバをキャッシュ・サイズ 12 GB で起動し、データベース *c:¥test¥mydemo.db* をロードします。

```
dbeng10 -n myserver -c 12G -cw c:¥test¥mydemo.db
```

-dt サーバ・オプション

テンポラリ・ファイルを保存するディレクトリを指定します。

構文

{ **dbsrv10** | **dbeng10** } **-dt** *temp-file-dir* ...

適用対象

すべてのサーバおよびオペレーティング・システム (UNIX 上の共有メモリ接続を除く)

備考

-dt オプションでは、テンポラリ・ファイルを保存するディレクトリを指定できます。このオプションを指定しないでデータベース・サーバを起動すると、SQL Anywhere は以下の環境変数をこの通りの順序で調べ、テンポラリ・ファイルの保存先とするディレクトリを決定します。

- SATMP
- TMP
- TMPDIR
- TEMP

これらの環境変数がいずれも定義されていない場合、テンポラリ・ファイルは、Windows の場合は現在のディレクトリ、UNIX の場合は */tmp* ディレクトリに作成されます。

UNIX の共有メモリ接続では、テンポラリ・ファイルは -dt で指定されたディレクトリには保存されません。環境変数がチェックされ、いずれの環境変数も定義されていない場合は */tmp* が使用されます。

参照

- 「データベース・ファイルの概要」 292 ページ
- 「SATMP 環境変数」 314 ページ
- 「異なるファイルの異なるデバイスへの配置」 『SQL Anywhere サーバ - SQL の使用法』

-ec サーバ・オプション

トランスポート・レイヤ・セキュリティまたは暗号化を使用して、すべてのクライアントとの間で転送される SQL Anywhere のすべてのネイティブ・パケット (DBLib、ODBC、OLE DB) を暗号化します。TDS パケットは暗号化されません。

構文

{ dbsrv10 | dbeng10 } -ec *encryption-options* …

encryption-options :

{ NONE |
SIMPLE |
TLS (TLS_TYPE=*cipher***;**
[FIPS={ Y | N };]
CERTIFICATE=*server-identity-filename***;**
CERTIFICATE_PASSWORD=*password* **) }**, …

備考

このオプションは、トランスポート・レイヤ・セキュリティを使用してクライアント・アプリケーションとデータベース・サーバ間の通信パケットを安全化する場合に使用します。「トランスポート・レイヤ・セキュリティ」 949 ページを参照してください。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

-ec オプションを指定すると、データベース・サーバは指定された暗号化タイプによって暗号化される接続のみ受け入れます。TDS プロトコルを介した接続は、jConnect を使用する Java アプリケーションを含みますが、-ec オプションの使用に関係なく常に受け入れられ、暗号化されることはありません。この TDS プロトコル・オプションを NO に設定すると、これらの暗号化されていない TDS 接続は禁止されます。「TDS プロトコル・オプション」 285 ページを参照してください。

デフォルトでは、通信パケットは暗号化されないため、セキュリティに潜在的なリスクがあります。ネットワーク・パケットのセキュリティが心配な場合は、-ec オプションを使用します。暗号化がパフォーマンスに及ぼす影響はごくわずかです。-ec オプションはサーバの暗号化設定を制御します。次のパラメータの 1 つ以上をカンマで区切ったリストで指定してください。

- **NONE** 暗号化されない接続を受け入れます。
- **SIMPLE** 単純暗号化された接続を受け入れます。このタイプの暗号化は、以前のバージョンの SQL Anywhere も含め、すべてのプラットフォームでサポートされます。単純暗号化では、サーバ認証、強力な楕円曲線暗号化、RSA 暗号化、その他のトランスポート・レイヤ・セキュリティ機能は提供されません。
- **TLS** 暗号化される接続を受け入れます。TLS パラメータに指定できる必須の引数は次のとおりです。
  - ***cipher*** RSA 暗号化の場合は **RSA** を指定し、ECC 暗号化の場合は **ECC** を指定します。FIPS 承認の RSA 暗号化の場合は、**TLS_TYPE=RSA;FIPS=Y** を指定します。RSA FIPS は別の承認ライブラリを使用しますが、SQL Anywhere 9.0.2 以降で RSA を指定しているクライアントと互換性があります。

    FIPS がサポートされているプラットフォームのリストについては、「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」の SQL Anywhere、Ultra Light、Mobile Link の各表にある「別途ライセンスが必要なコンポーネント」を参照してください。

    cipher は、証明書を作成するときに使用される暗号化 (ECC または RSA) と一致する必要があります。

    FIPS 承認のアルゴリズムの実行については、「-fips サーバ・オプション」 166 ページを参照してください。

    注意
    バージョン 10 クライアントでは、ECC アルゴリズムを使用してバージョン 9.0.2 以前のデータベース・サーバに接続することはできません。この構成で強力な暗号化が必要な場合は、RSA アルゴリズムを使用してください。

  - ***server-identity-filename*** サーバ ID のパスとファイル名を指定します。FIPS 承認の RSA 暗号化を使用している場合は、RSA 暗号化を使用して証明書を生成する必要があります。

サーバ証明書の作成については、「デジタル証明書の作成」 955 ページを参照してください。サーバ証明書は、自己署名証明書、または認証局やエンタープライズ・ルート証明書の署名を受けた証明書のいずれかです。

- ***password*** サーバのプライベート・キーのパスワードを指定します。このパスワードは、サーバ証明書を作成するときに指定します。

データベース・サーバが単純暗号化を受け入れ、暗号化されない接続を受け入れない場合、暗号化を使用しない TDS 接続以外の接続では、単純暗号化が使用されます。

-ec SIMPLE を指定してデータベース・サーバを起動すると、サーバは単純暗号化を使用した接続だけを受け入れます。TLS 接続 (ECC、RSA、RSA FIPS) は失敗し、暗号化を要求しない接続では単純暗号化が使用されます。

-ec SIMPLE,TLS(TLS_TYPE=ECC) を指定してサーバを起動すると、サーバは ECC 暗号化*または*単純暗号化を使用する接続だけを受け入れます。RSA 接続と RSA FIPS 接続はいずれも失敗し、暗号化を要求しない接続では単純暗号化が使用されます。

dbecc10.dll ファイルと *dbrsa10.dll* ファイルには、暗号化と復号化に使用される ECC コードと RSA コードが保存されています。*dbfips10.dll* には FIPS 承認の RSA アルゴリズムのコードがあります。データベース・サーバに接続するときに、適切なファイルが見つからなかったり、エラーが発生したりすると、サーバ・メッセージ・ウィンドウにメッセージが表示されます。指定されたタイプの暗号化を開始できない場合、サーバは起動しません。

クライアントとサーバで暗号化の設定が一致していることが必要です。設定が異なっていると、次の場合を除き、接続は失敗します。

- データベース・サーバに対して -ec SIMPLE を指定し、-ec NONE を指定しなかった場合、暗号化を要求しない接続は許可され、自動的に単純暗号化が使用されます。
- データベース・サーバ側で RSA を指定し、クライアント側で FIPS を指定している場合、またはその逆の場合には、接続は成功します。この場合、Encryption 接続プロパティはデータベース・サーバ側で指定された値を返します。

参照

- 「トランスポート・レイヤ・セキュリティを使用するデータベース・サーバの起動」 962 ページ
- 「Encryption 接続パラメータ [ENC]」 247 ページ
- 「-ek データベース・オプション」 222 ページ
- 「-ep サーバ・オプション」 164 ページ
- 「DatabaseKey 接続パラメータ [DBKEY]」 242 ページ

例

次の例は、暗号化されない接続と単純暗号化を使用する接続を許可します。

```
dbsrv10 -ec NONE,SIMPLE -x tcpip c:¥mydemo.db
```

次の例は、楕円曲線サーバ証明書 *sample.crt* を使用するデータベース・サーバを起動します。

```
dbsrv10 -ec TLS(TLS_TYPE=ECC;CERTIFICATE=sample.crt;CERTIFICATE_PASSWORD=tJ1#m6
+W) -x tcpip c:¥mydemo.db
```


次の例は、RSA サーバ証明書 *rsaserver.crt* を使用するデータベース・サーバを起動します。

```
dbsrv10 -ec TLS(TLS_TYPE=RSA;CERTIFICATE=rsaserver.crt;CERTIFICATE_PASSWORD=test) -x
tcpip c:¥mydemo.db
```

次の例は、FIPS 承認の RSA サーバ証明書 *rsaserver.crt* を使用するデータベース・サーバを起動します。

```
dbsrv10 -ec TLS
(TLS_TYPE=RSA;FIPS=Y;CERTIFICATE=rsaserver.crt;CERTIFICATE_PASSWORD=test) -x tcpip c:
¥mydemo.db
```

-ep サーバ・オプション

強力に暗号化されたデータベースを起動するときに、暗号化キーをユーザに要求します。

構文

{ dbsrv10 | dbeng10 } -ep …

備考

-ep オプションを指定すると、データベース・サーバは、コマンド・ラインから起動するときに暗号化キーの入力が必要なデータベースについて、ユーザが暗号化キーを入力するためのダイアログ・ボックスを表示します。このサーバ・オプションでは、クリア・テキストで暗号化キーを見ることができないようにすることで、高いセキュリティが得られます。

このオプションを指定した場合、ユーザが暗号化キーの入力を要求されるのは、次の条件がすべて該当する場合だけです。

- -ep オプションが指定されている
- サーバが Windows パーソナル・サーバであるか、サーバが起動したばかりである
- データベースを起動するキーが必要である
- サーバが Windows サービスではない、またはデスクトップとの対話オプションがオンになった Windows サービスである
- サーバがデーモンではない (UNIX)。

クライアント・アプリケーションとデータベース・サーバ間の通信パケットを安全化するには、-ec サーバ・オプションとトランスポート・レイヤ・セキュリティを使用します。「トランスポート・レイヤ・セキュリティ」 949 ページを参照してください。

参照

- 「トランスポート・レイヤ・セキュリティを使用するデータベース・サーバの起動」 962 ページ
- 「-ec サーバ・オプション」 161 ページ
- 「Encryption 接続パラメータ [ENC]」 247 ページ
- 「-ek データベース・オプション」 222 ページ
- 「DatabaseKey 接続パラメータ [DBKEY]」 242 ページ

例

myencrypted.db データベースが起動すると、ユーザは暗号化キーの入力を要求されます。

```
dbsrv10 -ep -x tcpip myencrypted.db
```

-fc サーバ・オプション

ファイル・システム・フルのコールバック関数がある DLL (UNIX では共有オブジェクト) のファイル名を指定します。

構文

{ **dbsrv10** | **dbeng10** } **-fc** *filename* …

適用対象

すべてのオペレーティング・システムとサーバ

備考

このオプションを使用すると、ファイル・システム・フルの状態が発生したときに、ユーザに通知して、必要に応じて修正措置をとることができます。-fc オプションを使用した場合、起動時にデータベース・サーバは指定の DLL の読み込みとコールバック関数のエントリ・ポイントの解決を試みます。DLL とエントリ・ポイントの両方を検出できない場合、SQL Anywhere データベース・サーバはエラーを返し、停止します。ユーザが提供する DLL は、コールバックを使用して指定のバッチ・ファイル (UNIX の場合はシェル・スクリプト) を呼び出し、診断または修正措置をとることができます。また、コールバック関数自体でもそのようなアクションを実行できます。

samples-dir¥SQLAnywhere¥DiskFull にディスク・フルのコールバック関数のサンプルがあります。

samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

SQL Anywhere は、他の DLL やファイルを検索する場合と同じロケーションでコールバック関数 DLL を検索します。

SQL Anywhere がファイルを検索する場所の詳細については、「SQL Anywhere のファイル検索方法」 325 ページを参照してください。

データベース・サーバは、ディスク・フルの状態を検出すると、コールバック関数 (指定されている場合) を呼び出し、それに以下の情報を渡します。

- 条件がトリガされた DB 領域の名前
- 失敗した操作からのオペレーティング・システム固有のエラー・コード

xp_out_of_disk からのリターン・コードにより、原因となった操作を中止するか再実行するかが決定されます。ゼロ以外の値が返された場合は、操作は中止され、それ以外の場合は操作が再実行されます。ゼロが返され、ファイル・システムの操作が失敗するかぎり、コールバック関数が繰返し呼び出されます。

Windows プラットフォームでは、データベース・サーバがサーバ・メッセージ・ウィンドウで開始され (-qi も -qw も指定されていない)、コールバック DLL が指定されていない場合は、ディス

ク・フルの状態が発生するとダイアログが表示されます。このダイアログには、DB 領域名とエラー・コードが含まれているので、ユーザはディスク・フルの状態の原因となった操作を再実行するか中止するかを決定できます。

その他すべてのオペレーティング・システムでは、-fc が指定されていない場合にディスク・フルの状態になると、致命的なエラーが発生します。

データベース・ファイル、ログ・ファイル、テンポラリ・ファイルなどを格納しているデバイスの使用可能なディスク領域を管理するシステム・イベントを作成して、ディスク領域が不足している場合に管理者に警告できます。

「CREATE EVENT 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

参照

- 「コールバック関数の使い方」 『SQL Anywhere サーバ - プログラミング』
- 「システム・イベントの概要」 874 ページ
- 「max_temp_space オプション [データベース]」 469 ページ
- 「temp_space_limit_check オプション [データベース]」 503 ページ

例

データベース・サーバは、起動時に *diskfull.dll* DLL をロードしようとします。

```
dbeng10 -fc diskfull.dll
```

-fips サーバ・オプション

強力なデータベースと通信の暗号化に FIPS 承認のアルゴリズムのみを使用することを要求します。

構文

{ **dbsrv10** | **dbeng10** } **-fips** …

備考

このオプションを指定すると、すべてのサーバ暗号化で FIPS 承認のアルゴリズムが使用されます。このオプションは、強力なデータベース暗号化、クライアント／サーバのトランスポート・レイヤ・セキュリティ、Web サービスのトランスポート・レイヤ・セキュリティに適用されます。-fips オプションが指定されているときには、暗号化されていない接続とデータベースは使用できますが、単純暗号化は使用できません。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

強力なデータベース暗号化では、-fips オプションを指定すると、CREATE DATABASE 文の ALGORITHM 句で AES が指定されている場合も、新しいデータベースは AES_FIPS タイプを使用します。データベース・サーバを -fips で起動した場合、AES または AES_FIPS の強力な暗号

化によって暗号化されたデータベースを実行できますが、単純暗号化で暗号化されたデータベースは実行できません。-fips が指定されている場合、暗号化されていないデータベースもサーバで開始できます。

AES_FIPS で暗号化したデータベースを実行するために使用するコンピュータには、SQL Anywhere セキュリティ・オプションをインストールしてください。

SQL Anywhere トランスポート・レイヤ・セキュリティでは、-fips オプションを指定すると、RSA が指定されていても、サーバは FIPS 承認の RSA 暗号化を使用します。ECC を指定した場合、FIPS 承認の楕円曲線アルゴリズムは使用できないため、エラーが発生します。

Web サービス用の トランスポート・レイヤ・セキュリティでは、-fips オプションを指定すると、HTTPS が指定されていてもサーバは HTTPS FIPS を使用します。

-fips を指定すると、ENCRYPT 関数と HASH 関数では FIPS 承認の RSA 暗号化が使用され、パスワード・ハッシングではアルゴリズムとして SHA-256 ではなく SHA-256 FIPS が使用されます。

参照

- 「強力な暗号化」 936 ページ
- 「トランスポート・レイヤ・セキュリティ」 949 ページ
- 「SQL Anywhere Web サービスでのトランスポート・レイヤ・セキュリティの使用」 966 ページ
- 「-ec サーバ・オプション」 161 ページ
- 「ENCRYPT 関数 [文字列]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「HASH 関数 [文字列]」 『SQL Anywhere サーバ - SQL リファレンス』

-ga サーバ・オプション

最後の接続が切断された後で、データベースをアンロードします。

構文

{ dbsrv10 | dbeng10 } -ga …

適用対象

NetWare を除くすべてのオペレーティング・システム

備考

ネットワーク・サーバ上でこのオプションを指定すると、最後の接続が切断された後で各データベースがアンロードされます。最後の接続が切断された後に各データベースをアンロードし、最後のデータベースが停止したときにサーバも停止します。

-gb サーバ・オプション

サーバ・プロセスの優先クラスを設定します。

構文

{ dbsrv10 | dbeng10 } -gb { idle | normal | high | maximum } …

適用対象

Windows

備考

サーバ・プロセスの優先クラスを設定します。値 idle は万全を期すために用意されており、maximum はコンピュータの実行に影響を及ぼす可能性があります。一般的な設定として使用されるのは、normal または high です。

-gc サーバ・オプション

チェックポイント間の期待最大間隔を設定します。

構文

{ **dbsrv10** | **dbeng10** } **-gc** *integer* …

適用対象

すべてのオペレーティング・システムとサーバ

備考

各データベースでチェックポイントを行わずに、データベース・サーバを実行する期待*最大*時間を分数で設定します。

デフォルト値は 60 分です。値 0 を入力すると、デフォルト値の 60 が使用されます。

通常、チェックポイントは、指定する時間より頻繁に発生します。

「データベース・サーバがチェックポイントのタイミングを決定する方法」 842 ページを参照してください。

参照

- 「checkpoint_time オプション [データベース]」 432 ページ
- 「チェックポイントとチェックポイント・ログ」 835 ページ

-gd サーバ・オプション

データベースを起動または停止するために必要なパーミッションを設定します。

構文

{ **dbsrv10** | **dbeng10** } **-gd** { **DBA** | **all** | **none** } …

適用対象

すべてのオペレーティング・システムとサーバ

備考

このパーミッションは、ユーザが新しいデータベース・ファイルをサーバにロードするために、または実行中のデータベース・サーバでデータベースを停止するために必要なパーミッションです。次のいずれかのレベルを指定します。

- **DBA**　DBA 権限のあるユーザだけがデータベースを起動または停止できます。
- **all**　すべてのユーザがデータベースを起動または停止できます。
- **none**　データベース・サーバが起動しているか停止しているかに関係なく、データベースの開始と停止は許可されません。

パーソナル・データベース・サーバのデフォルト設定は all で、ネットワーク・データベース・サーバのデフォルト設定は DBA です。大文字と小文字の両方の構文を使用できます。

このオプションを DBA に設定した場合、データベースを開始または停止させるためにクライアント・アプリケーションがサーバに接続されていなければならないことに注意してください。新しい接続で DBA ユーザ ID とパスワードを指定するだけでは不十分です。

次のように StartDBPermission サーバ・プロパティを使用して -gd オプションの設定を取得できます。

```
SELECT PROPERTY ( 'StartDBPermission' )
```

例

ネットワーク・データベース・サーバで -gd オプションを使用する手順は、次のとおりです。

1. ネットワーク・データベース・サーバを起動します。

   ```
   dbsrv10 -x tcpip -su mypwd -n myserver -gd DBA
   ```

2. Interactive SQL からユーティリティ・データベースに接続します。

   ```
   dbisql -c "UID=DBA;PWD=mypwd;ENG=myserver;DBN=utility_db"
   ```

3. データベースを起動します。

   ```
   START DATABASE demo
   ON myserver;
   ```

4. 起動したデータベースに接続します。

   ```
   CONNECT
   TO myserver
   DATABASE demo
   USER DBA IDENTIFIED BY sql
   ```

-ge サーバ・オプション

外部関数のスタック・サイズを設定します。

構文

{ **dbsrv10** | **dbeng10** } **-ge** *integer* …

適用対象

Windows、NetWare

備考

外部関数を実行するスレッドのスタック・サイズをバイト数で設定します。デフォルトは 32 KB です。

-gf サーバ・オプション

サーバ上でトリガを起動不可にします。

構文

{ dbsrv10 | dbeng10 } -gf …

適用対象

すべてのオペレーティング・システムとサーバ

備考

-gf サーバ・オプションは、トリガの起動を無効にするようにサーバに指示します。

参照

- 「fire_triggers オプション [互換性]」 450 ページ

-gk サーバ・オプション

dbstop を使用して、ネットワーク・サーバとパーソナル・サーバを停止するために必要なパーミッションを設定します。

構文

{ dbsrv10 | dbeng10 } -gk { DBA | all | none } …

適用対象

すべてのオペレーティング・システムとサーバ

備考

指定できる値は次のとおりです。

- **DBA** DBA 権限を持つユーザだけが dbstop を使ってサーバを停止できます。これはネットワーク・サーバのデフォルトです。
- **all** すべてのユーザが dbstop を使ってサーバを停止できます。これはパーソナル・サーバのデフォルトです。
- **none** dbstop を使ってサーバを停止できません。

大文字と小文字の両方の構文を使用できます。

-gl サーバ・オプション

LOAD TABLE を使用するデータのロード、UNLOAD または UNLOAD TABLE を使用するデータのアンロードに必要なパーミッションを設定します。

構文

{ dbsrv10 | dbeng10 } -gl { DBA | all | none } ⋯

適用対象

すべてのオペレーティング・システムとサーバ

備考

UNLOAD TABLE や UNLOAD 文は、データベース・サーバ・コンピュータのファイルにデータを配置します。LOAD TABLE 文は、データベース・サーバ・コンピュータからファイルを読み込みます。

これらの文を使用してファイル・システムへのアクセスを制御するには、-gl サーバ・オプションを使用します。このオプションによって、これらの文を使用するために必要なデータベース・パーミッションのレベルを制御できます。

使用できる値は次のとおりです。

- **DBA** DBA 権限を持つユーザだけが、データベースからデータをロード／アンロードできます。
- **all** すべてのユーザがデータベースからデータをロード／アンロードできます。
- **none** データのロードやアンロードはできません。

大文字と小文字の両方の構文を使用できます。

UNIX 以外のオペレーティング・システムを使用するパーソナル・データベース・サーバの場合、デフォルト設定は all です。ネットワーク・データベース・サーバや UNIX パーソナル・サーバの場合、デフォルト設定は DBA です。これらの設定は、UNIX 以外のプラットフォームではパーソナル・データベース・サーバが現在のコンピュータで実行され、したがってユーザはすでにファイル・システムにアクセスしているという事実を反映しています。

-gm サーバ・オプション

サーバに対する同時接続の数を制限します。

構文

{ dbsrv10 | dbeng10 } -gm *integer* ⋯

適用対象

すべてのオペレーティング・システムとサーバ

備考

サーバの接続制限を定義します。ライセンス契約に許可されている数より大きい値、もしくはメモリ制約を超えた値をここで設定した場合、その値は無効です。

緊急時に DBA がサーバに接続して他の接続を削除できるように、データ・サーバは、接続制限を超えて DBA 接続を 1 つ追加して許可します。

-gn サーバ・オプション

データベース・サーバが同時に実行できるタスクの最大数を設定します。

構文

{ **dbsrv10** | **dbeng10** } **-gn** *integer* …

適用対象

すべてのオペレーティング・システムとサーバ

備考

このオプションは、データベース・サーバの最大マルチプログラミング・レベルを設定します。データベース・サーバが同時に実行できるタスク (ユーザ要求とシステム要求) の数を制限します。データベース・サーバが最大数を超える要求を受け取った場合、新しい要求は実行中のタスクが完了するまで待つことになります。

未スケジュールの要求とアクティブな要求を加算した総数は、-gm サーバ・オプションで制限されます。これにより、サーバへの接続数を制限します。

ネットワーク・データベース・サーバとパーソナル・データベース・サーバではいずれも、アクティブ・タスクのデフォルト値は 20 に設定されています。ただし、Windows CE の場合は、このデフォルト値は 3 であり、同時に実行できるアクティブ・タスクの数は、データベース・サーバのスレッドの数と、使用されている論理プロセッサの数によって異なります。

データベース・サーバのカーネルは、スケジューリングの単位としてタスクを使用します。ユーザ要求を実行するには、最低 1 つのタスクが必要です。ただし、1 つの要求に伴って追加のタスクがスケジューリングされることがあります。たとえば、Java のストアド・プロシージャや関数の実行を伴う要求では、データベース・サーバに対してデータベース要求が発行されます。

クエリ内並列処理が関連する場合は、同時に実行される各アクセス・プラン・コンポーネントがタスクになります。これらのタスクは、実際の個別の要求であるものとして、-gn オプションで指定された制限に考慮されます。ただし、クエリ内並列処理で作成されたタスクは、アクティブな要求とアクティブでない要求の数を追跡するデータベース・プロパティには反映されません。

参照

- 「SQL Anywhere でのスレッド」 23 ページ
- 「データベース・サーバのマルチプログラミング・レベルの設定」 27 ページ
- 「max_query_tasks オプション [データベース]」 466 ページ
- 「-gm サーバ・オプション」 171 ページ
- 「-gm サーバ・オプション」 171 ページ
- 「-gtc サーバ・オプション」 175 ページ

-gp サーバ・オプション

許可される最大データベース・ページ・サイズを設定します。

構文

{ dbsrv10 | dbeng10 } -gp { 2048 | 4096 | 8192 | 16384 | 32768 } …

適用対象

すべてのオペレーティング・システムとサーバ

備考

サーバのページ・サイズよりも大きいページ・サイズのデータベース・ファイルはロードできません。このオプションは、サーバのページ・サイズをバイト数で明示的に設定します。

このオプションを指定しないと、コマンド・ラインに指定された最初のデータベースのページ・サイズが使用されます。

いずれのプラットフォームでも、このオプションを指定せずにデータベースがロードされていない状態でサーバを起動した場合、デフォルト値は 4096 になります。

-gr サーバ・オプション

システム障害からのリカバリに要する最長時間を分数で設定します。

構文

{ dbsrv10 | dbeng10 } -gr *integer* **…**

適用対象

すべてのオペレーティング・システムとサーバ

備考

データベース・サーバが複数のデータベースで実行されている場合、このオプションで書き換えないかぎり、最初に起動されたデータベースで指定されているリカバリ時間が使用されます。

-gr オプションで指定した値は、データベース・サーバにチェックポイントを実行する頻度を指示します。たとえば、-gr を 5 に設定した場合、データベース・サーバは頻繁にチェックポイントを実行して、リカバリが 5 分以上かからないようにします。ただし、リカバリが必要な場合は、-gr で指定した時間より長くかかったとしても、完了するまで実行されます。

リカバリ時間には、データベースのリカバリ予想時間とチェックポイント予想時間が含まれます。

参照

- 「recovery_time オプション [データベース]」 488 ページ

-gss サーバ・オプション

サーバの内部実行スレッドあたりのスタック・サイズを設定します。

構文

{ **dbsrv10** | **dbeng10** } **-gss** { *integer*[**k** | **m**] } …

適用対象

このオプションは、Windows オペレーティング・システムでは無効です。

備考

内部実行スレッドの数は、-gn オプションにより制御されます。デフォルト値は 20 です。メモリが限られている環境では、-gss オプションを使用してデータベース・サーバのメモリ使用量を減らすことができます。

size には、使用するメモリ容量をバイト単位で指定します。キロバイトまたはメガバイトの単位を指定するには、それぞれ **k**、**m** を使用します。

NetWare では、内部実行スレッドあたりのデフォルトおよび最小のスタック・サイズは 128 KB で、最大スタック・サイズは 1 MB です。UNIX では、内部実行スレッドあたりのデフォルトおよび最小のスタック・サイズは 500 KB で、最大スタック・サイズは 4 MB です。

参照

- 「SQL Anywhere でのスレッド」 23 ページ

-gt サーバ・オプション

使用できる物理プロセッサの最大数を設定する (ライセンスされたプロセッサの数を上限とする)。このオプションは、マルチプロセッサ・システムでのみ役立ちます。

構文

{ **dbsrv10** | **dbeng10** } **-gt** *integer* …

適用対象

Windows (Windows CE を除く)、Linux、Solaris

備考

per-seat ライセンスの場合、ネットワーク・データベース・サーバはコンピュータで使用可能なすべての CPU を使用します (デフォルト)。CPU ベースのライセンスの場合、ネットワーク・データベース・サーバはライセンスを受けたプロセッサ数のみ使用可能です。また、パーソナル・データベース・サーバとランタイム・データベース・サーバは、いずれも 1 つのプロセッサしか使用できません。

-gt オプションの値を指定すると、データベース・サーバは関係マスクを調整し (ハードウェア・プラットフォームでサポートされている場合)、指定した物理プロセッサの数のみを使用して実行するようにデータベース・サーバを制限します。データベース・サーバが *n* 個のプロセッサのライセンスを受けている場合、デフォルトでは、サーバは *n* 個の物理プロセッサにおいて、すべ

ての論理プロセッサ (ハイパースレッドとコア) 上で実行します。-gtc オプションを使用すると、さらに制限を加えることができます。

-gt オプションには、1 と、以下に挙げる項目の最小値の間で値を設定できます。

- コンピュータ上の物理プロセッサの数
- サーバがライセンスを受けている CPU の最大数 (CPU がライセンスされている場合)

-gt オプションに範囲外の値を指定すると、下限または上限の値が設定されます。パーソナル・データベース・サーバ (dbeng10) では、サーバは -gt オプションの値として 1 を使用します。

参照

- 「-gn サーバ・オプション」 172 ページ
- 「-gtc サーバ・オプション」 175 ページ
- 「SQL Anywhere でのスレッド」 23 ページ

-gtc サーバ・オプション

データベース・サーバが許容するプロセッサ同時実行性の最大値を指定します。

構文

{ **dbsrv10** | **dbeng10** } **-gtc** *logical-processors-to-use* …

適用対象

Linux、Solaris、Windows オペレーティング・システム (Intel 互換の x86 と x64 プラットフォームを実行するシステム、ただし、Windows CE を除く)

備考

データベース・サーバを起動すると、検出された物理プロセッサと論理プロセッサの数がサーバ・メッセージ・ウィンドウに表示されます。

物理プロセッサは、コンピュータの CPU であり、「**パッケージ**」または「**ダイ**」と呼ばれることもあります。物理プロセッサがハイパースレッドをサポートする場合や、**マルチプロセッサ** (一般に「マルチコア・プロセッサ」と呼ばれます) として設定されている場合は、追加の論理プロセッサが存在します。オペレーティング・システムは、論理プロセッサ上でスレッドをスケジュールします。

-gtc オプションでは、データベース・サーバが使用する論理プロセッサの数を指定できます。このオプションを指定することで、サーバの起動時に作成されるデータベース・サーバ・スレッドの数を制限します。これは、データベース・サーバで同時に実行できるアクティブ・タスクの数を制限することになります。デフォルトでは、作成されるスレッドの数は (1 + ライセンスを受けているすべての物理プロセッサ上にある論理プロセッサの数) です。

デフォルトでは、ライセンスされた各物理プロセッサにおいて、すべての論理プロセッサ (コアまたはハイパースレッド) の同時使用が可能です。たとえば、1 つの CPU を搭載し、ハイパースレッドをサポートするシステムについて考えてみます。データベース・サーバは、デフォルトで、1 つの物理プロセッサ上で同時に実行する 2 つのスレッドを許可します。-gtc オプションが指定されている場合で、使用される論理プロセッサの数が、ライセンスされている物理プロセッ

サで利用できる総数よりも少なくなると、データベース・サーバはラウンド・ロビン方式に基づいて論理プロセッサを割り付けます。-gtc オプションに対して暗黙的に 1 を指定すると、クエリ内並列処理 (クエリの並列処理) は無効になります。クエリ内並列処理は、max_query_tasks オプションを使用して、明示的に制限したり、無条件に無効にしたりすることもできます。
「max_query_tasks オプション [データベース]」 466 ページを参照してください。

参照

- 「-gn サーバ・オプション」 172 ページ
- 「-gt サーバ・オプション」 174 ページ
- 「クエリ実行時の並列処理」 『SQL Anywhere サーバ - SQL の使用法』
- 「SQL Anywhere でのスレッド」 23 ページ

例

Windows ベースの SMP コンピュータを例に挙げて考えてみます。ここでは、システムが 4 プロセッサ構成で、各物理プロセッサにコアが 2 つある (したがって、8 つの論理プロセッサがある) ことを前提としています。物理プロセッサは文字で、論理プロセッサ (この例では、コア) は数字でそれぞれ区別します。このシステムには 4 プロセッサ・ユニットとして A0、A1、B0、B1、C0、C1、D0、D1 が存在することになります。

シナリオ	ネットワーク・データベース・サーバの設定
シングル CPU ライセンス、または -gt に 1 が指定されている	◆ -gt 1 ◆ -gtc 2 ◆ -gn 20 スレッドは A0 または A1 で実行できます。
CPU のライセンスに制限がなく、-gtc に 5 が指定されている	◆ -gt 4 ◆ -gtc 5 ◆ -gn 20 スレッドは A0、A1、B0、C0、D0 で実行できます。
データベース・サーバに 3 CPU ライセンスがあり、-gtc 5 が指定されている	◆ -gt 3 ◆ -gtc 5 ◆ -gn 20 スレッドは A0、A1、B0、B1、C0 で実行できます。
CPU のライセンスに制限がなく、-gtc に 1 が指定されている	◆ -gt 4 ◆ -gtc 1 ◆ -gn 20 スレッドは A0 でのみ実行できます。

-gu サーバ・オプション

ユーティリティ・コマンドのパーミッション・レベルを設定します。

構文

{ dbsrv10 | dbeng10 } -gu { all | none | DBA | utility_db } ⋯

適用対象

すべてのオペレーティング・システムとサーバ

備考

CREATE DATABASE や DROP DATABASE などのユーティリティ・コマンドのパーミッション・レベルを設定します。このレベルは、utility_db、all、none、DBA のいずれかに設定できます。デフォルトは DBA です。

utility_db レベルでは、これらのコマンドをユーティリティ・データベースに接続できるユーザだけが使用できます。ユーティリティ・コマンドの実行は、all レベルではすべてのユーザが許可され、none レベルではすべてのユーザが不許可となり、DBA レベルでは DBA 権限を持つユーザが許可されます。

-k サーバ・オプション

パフォーマンス・モニタ統計値の収集を制御します。

構文

{ dbsrv10 | dbeng10 } -k ...

適用対象

すべてのオペレーティング・システムとサーバ

備考

データベース・サーバは、デフォルトでパフォーマンス・モニタ統計値を収集します。

データベース・サーバの起動時に -k を指定すると、パフォーマンス・モニタ統計値は収集されません。-k オプションは、クエリ・オプティマイザが使用するカラム統計には影響しません。

このオプションは、データベース・サーバをマルチプロセッサ・コンピュータで実行しており、テストによってパフォーマンスの向上が確認できた場合のみ使用してください。通常の負荷レベルでは、パフォーマンスの向上はわずかであるため、このオプションの使用は推奨できません。パフォーマンス・カウンタを無効にすると、パフォーマンス上の問題が発生した場合、分析に必要な情報を取得できません。

パフォーマンス・モニタ統計値の収集についての設定は sa_server_option システム・プロシージャでも変更できます。「sa_server_option システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

参照

- 「Sybase Central パフォーマンス・モニタを使用したモニタリング」 『SQL Anywhere サーバ - SQL の使用法』

-kl サーバ・オプション

Kerberos GSS-API ライブラリ (UNIX では共有オブジェクト) のファイル名を指定し、データベース・サーバへの Kerberos 認証接続を有効にします。

構文

{ **dbsrv10** | **dbeng10** } **-kl** *GSS-API-library-file* ...

適用対象

NetWare と WindowsCE を除くすべてのオペレーティング・システム

備考

このオプションでは、Kerberos GSS-API のロケーションと名前を指定します。このオプションが必要となるのは、Kerberos クライアントでデフォルトと異なる Kerberos GSS-API ライブラリ・ファイル名が使用されているか、データベース・サーバを実行しているコンピュータに複数の GSS-API ライブラリがインストールされている場合だけです。Kerberos クライアントのインストールと設定が完了し、データベース・サーバが SSPI を使用できない状態であることが必要です。

このオプションを指定すると、データベース・サーバに対する Kerberos 認証が有効になります。

参照

- 「-kr サーバ・オプション」 178 ページ
- 「-krb サーバ・オプション」 179 ページ
- 「Kerberos 接続パラメータ [KRB]」 253 ページ
- 「Kerberos 認証の使用」 108 ページ
- 「GRANT 文」 『SQL Anywhere サーバ - SQL リファレンス』

例

次のコマンドは、Kerberos 認証に *libgssapi_krb5.so* 共有オブジェクトを使用するデータベース・サーバを起動します。

```
dbsrv10 -kl libgssapi_krb5.so -n my_server_princ /opt/myapp/kerberos.db
```

-kr サーバ・オプション

Kerberos サーバ・プリンシパルの領域を指定し、データベース・サーバへの Kerberos 認証接続を有効にします。

構文

{ **dbsrv10** | **dbeng10** } **-kr** *server-realm* ...

適用対象

NetWare と WindowsCE を除くすべてのオペレーティング・システム

備考

このオプションでは、Kerberos サーバ・プリンシパルの領域を指定します。通常は、データベース・サーバが Kerberos 認証に使用するプリンシパルは *server-name@default-realm* です。*default-realm* は Kerberos クライアントに設定されたデフォルト領域です。このオプションは、サーバ・プリンシパルの領域をデフォルト領域以外に変更する場合に使用します。このオプションを指定すると、サーバ・プリンシパルは *server-name@server-realm* となります。

このオプションを指定すると、データベース・サーバに対する Kerberos 認証が有効になります。

参照

- 「-kl サーバ・オプション」 178 ページ
- 「-krb サーバ・オプション」 179 ページ
- 「Kerberos 接続パラメータ [KRB]」 253 ページ
- 「Kerberos 認証の使用」 108 ページ
- 「GRANT 文」『SQL Anywhere サーバ - SQL リファレンス』

例

次のコマンドは、Kerberos ログインを受け入れて認証にプリンシパル my_server_princ@MYREALM を使用するデータベース・サーバを起動します。

```
dbeng10 -kr MYREALM -n my_server_princ C:¥kerberos.db
```

-krb サーバ・オプション

データベース・サーバへの Kerberos 認証接続を有効にします。

構文

{ **dbsrv10** | **dbeng10** } **-krb** ...

適用対象

NetWare と WindowsCE を除くすべてのオペレーティング・システム

備考

このオプションは、データベース・サーバに対する Kerberos 認証を有効にします。データベース・サーバが Kerberos を使用してクライアントを認証するためには、-krb、-kl、-kr のうち 1 つ以上を指定する必要があります。

Kerberos 認証を使用するためには、クライアント・コンピュータとデータベース・サーバ・コンピュータの両方に Kerberos クライアントがインストールされ、設定が完了している必要があります。さらに、Kerberos KDC にプリンシパル *server-name@REALM* があり、プリンシパル *server-name@REALM* のキータブがデータベース・サーバ・コンピュータ上の keytab ファイルに抽出されていることも必要です。この準備が完了していない場合、-krb オプションを指定すると、データベース・サーバは起動しません。

注意
データベース・サーバ名には、/、¥、@ を使用できません。また、Kerberos ではマルチバイト文字を使用したデータベース・サーバ名は使用できません。

Kerberos ログインを許可するためには login_mode データベース・オプションを設定する必要があります。また、GRANT KERBEROS LOGIN 文を使用して、Kerberos クライアント・プリンシパルをデータベース・ユーザ ID にマップすることも必要です。

参照

- 「-kl サーバ・オプション」 178 ページ
- 「-kr サーバ・オプション」 178 ページ
- 「Kerberos 接続パラメータ [KRB]」 253 ページ
- 「Kerberos 認証の使用」 108 ページ
- 「GRANT 文」 『SQL Anywhere サーバ - SQL リファレンス』

例

次のコマンドは、データベース・サーバの Kerberos プリンシパル my_server_princ@MYREALM に対し、データベース・サーバ my_server_princ を起動します。

```
dbsrv10 -krb -n my_server_princ C:¥kerberos.db
```

-m サーバ・オプション

チェックポイントの実行後にトランザクション・ログの内容を削除します。

構文

{ dbsrv10 | dbeng10 } -m …

適用対象

すべてのオペレーティング・システムとサーバ

備考

このオプションは、シャットダウン時、またはサーバでスケジュールされたチェックポイントの結果としてチェックポイントが実行されたときに、トランザクション・ログの内容を削除します。

警告
このオプションを選択すると、データベース・ファイルを含むデバイスのメディア障害に対して無防備な状態になります。

これでトランザクション・ログの肥大化が自動的に制限されます。チェックポイントの頻度は、checkpoint_time と recovery_time オプションによって制御できます (また、コマンド・ラインでも設定できます)。

-m オプションは、高速な応答時間を必要とする大容量のトランザクションを処理する場合や、リカバリやレプリケーションがトランザクション・ログの内容に依存しない場合に、トランザク

ション・ログのサイズを制限するのに役立ちます。-m オプションは、各 COMMIT の後にチェックポイントが必要で、その結果パフォーマンスが低下するような場合に、トランザクション・ログなしで稼働する場合の代替策となります。-m オプションを指定すると、データベース・ファイルを含むデバイスのメディア障害に対して無防備な状態になります。-m オプションを使用する前に、トランザクション・ログを管理する他の代替策 (BACKUP 文やイベントの使用など) を検討してください。

データベース・ファイルの断片化を防ぐためには、このオプションを使用する場合に、トランザクション・ログをデータベースそのものとは別のデバイスまたはパーティションに保管することをおすすめします。

このオプションを指定すると、チェックポイントの実行中に他の操作は行われません。

レプリケートまたは同期されるデータベース
レプリケートされるデータベースまたは同期されるデータベースでは、-m オプションを使わないでください。SQL Remote と Mobile Link で使用されるレプリケーションと同期は、本質的にトランザクション・ログ情報に依存します。

-n サーバ・オプション

データベース・サーバの名前を設定します。

構文

{ dbsrv10 | dbeng10 } -n *database-filename…*

適用対象

すべてのオペレーティング・システムとサーバ

備考

デフォルトでは、データベース・サーバはパスと拡張子を除いた最初のデータベース・ファイル名を受け取ります。たとえば、*samples-dir¥demo.db* ファイルでサーバを起動するときに、-n オプションを指定しなかった場合、サーバの名前は demo になります。

起動時にはデータベースは照合されないので、サーバ名はコンピュータの文字セットに基づいて解釈されます。サーバ名にマルチバイト文字を使用することは推奨できません。

データベース・サーバの名前は、有効な識別子であることが必要です。データベース・サーバのロング・ネームは、プロトコルごとに異なる長さにトランケートされます。データベース・サーバには、以下に該当する名前を付けることはできません。

- 空白スペース、一重引用符、または二重引用符で始まる名前
- 空白スペースで終わる名前
- セミコロンを含む名前

プロトコル	トランケーションの長さ
TCP/IP	250 バイト
共有メモリ	250 バイト
SPX	32 バイト

注意
Windows と UNIX では、データベース・サーバがバージョン 10.0.0 以降で、名前が次の長さを超えている場合、バージョン 9.0.2 以前のクライアントから接続することはできません。

- Windows 共有メモリの場合は、40 バイト
- UNIX 共有メモリの場合は、31 バイト
- TCP/IP の場合は、40 バイト

サーバ名は、クライアント・アプリケーション接続文字列かプロファイルの EngineName (ENG) 接続パラメータで使用する名前を指定します。少なくとも 1 つのデータベース・サーバがコンピュータで実行されている場合、共有メモリ環境では、サーバ名を指定しなかったときに使用されるデフォルトのデータベース・サーバが存在します。

同じ名前で複数のデータベース・サーバを実行することはおすすめしません。

2 つの -n オプション
-n オプションは、指定された位置によって意味が変わります。データベース・ファイル名の前に指定すると、サーバ・オプションとしてサーバを指定します。データベース・ファイル名の後に指定すると、データベース・オプションとしてデータベースを指定します。
たとえば、次のコマンドでは、データベース・サーバ SERV とデータベース DATA を指定しています。

```
dbsrv10 -n SERV sales.db -n DATA
```

「-n データベース・オプション」 223 ページを参照してください。

参照

- 「識別子」 『SQL Anywhere サーバ - SQL リファレンス』
- 「EngineName 接続パラメータ [ENG]」 249 ページ

-o サーバ・オプション

サーバ・メッセージ・ウィンドウへのすべての出力をコンソール・ログに書き込みます。

構文

{ **dbsrv10** | **dbeng10** } **-o** *filename* …

適用対象

すべてのオペレーティング・システムとサーバ

備考

サーバ・メッセージ・ウィンドウへのすべての出力をサーバ・メッセージ・ウィンドウに表示するとともに、コンソール・ログにも書き込みます。-qi オプションを -o とともに指定すると、すべてのメッセージがコンソール・ログ・ファイルだけに出力されます。

次のコマンドを実行すると、コンソール・ログのファイル名を取得できます。

```
SELECT PROPERTY ( 'ConsoleLogFile' )
```

参照

- 「-os サーバ・オプション」 184 ページ
- 「-ot サーバ・オプション」 185 ページ
- 「-qi サーバ・オプション」 187 ページ

-oe サーバ・オプション

起動エラー、致命的なエラー、アサーションをロギングするファイルの名前を指定します。

構文

{ **dbsrv10** | **dbeng10** } **-oe** *filename* ...

適用対象

すべてのオペレーティング・システムとサーバ

備考

出力ログ・ファイルの各行の先頭には、日付と時刻が記録されます。起動エラーには、次のようなエラーがあります。

- データベース・ファイル *database file* が開けない／読み込めない
- その名前のデータベース・サーバがすでに起動している

致命的なエラーとアサーションは、-oe が指定されているかどうかにかかわらず、Windows アプリケーション・イベント・ログ (Window CE を除く) または UNIX システム・ログにログされます。

-on サーバ・オプション

コンソール・ログの最大サイズを指定します。ログ・ファイルがこのサイズに達すると、現在のファイルが拡張子 *.old* を持つ名前に変更され、新しいファイルが作成されます。

構文

{ **dbsrv10** | **dbeng10** } **-on** { *size*[**k** | **m** | **g**] } ...

備考

size には、コンソール・ログ・ファイルの最大サイズをバイト単位で指定します。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれサフィックス **k**、**m**、または **g** を使用します。最小サイズは 10 KB です。デフォルトでは、最大サイズは無制限となります。

コンソール・ログが指定されたサイズに達すると、出力ファイルが拡張子 *.old* を持つ名前に変更され、元の名前で新しいファイルが開始されます。

> **注意**
> 拡張子が *.old* であるコンソール・ログがすでに存在する場合は、そのファイルが上書きされます。古いコンソール・ログが削除されないようにするには、代わりに -os オプションを使用します。

このオプションは、-os オプションと同時に使用できません。

参照

- 「-o サーバ・オプション」 182 ページ
- 「-os サーバ・オプション」 184 ページ
- 「-ot サーバ・オプション」 185 ページ

-os サーバ・オプション

コンソール・ログの最大サイズを指定します (このサイズに達するとログのファイル名が変更されます)。

構文

{ **dbsrv10** | **dbeng10** } **-os** { *size*[**k** | **m** | **g**] } ...

適用対象

すべてのオペレーティング・システムとサーバ

備考

size には、コンソール・メッセージのログを書き込むファイルの最大サイズをバイト単位で指定します。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれサフィックス **k**、**m**、または **g** を使用します。最小サイズは 10 KB です。デフォルトでは、最大サイズは無制限となります。

データベース・サーバは、出力メッセージをコンソール・ログに書き込む前に、現在のファイル・サイズを確認します。新しいログ・メッセージを書き込むと、ファイルが指定されたサイズを超える場合は、コンソール・ログのファイル名が *yymmddxx.slg* に変更されます。*yymmdd* は、そのファイルが作成された年、月、日を表します。*xx* は 00 で始まる番号で、1 ずつ増えていきます。

このオプションによって、コンソール・ファイルが古いこと確認して削除し、ディスク領域を解放できます。

このオプションは、-os オプションとは一緒に使用できません。

参照

- 「-o サーバ・オプション」 182 ページ
- 「-on サーバ・オプション」 183 ページ
- 「-ot サーバ・オプション」 185 ページ

-ot サーバ・オプション

コンソール・ログをトランケートし、そのファイルに出力メッセージを追加します。

構文

{ **dbsrv10** | **dbeng10** } **-ot** *logfile* …

適用対象

すべてのオペレーティング・システムとサーバ

備考

このオプションの機能は、コンソール・ログのトランケート後にメッセージが書き込まれる点を除き、-o オプションと同じです。次のコマンドを実行すると、コンソール・ログの名前を取得できます。

```
SELECT PROPERTY ( 'ConsoleLogFile' )
```

参照

- 「-o サーバ・オプション」 182 ページ
- 「-os サーバ・オプション」 184 ページ

-p サーバ・オプション

通信パケットの最大サイズを設定します。

構文

{ **dbsrv10** | **dbeng10** } **-p** *integer* …

適用対象

すべてのオペレーティング・システムとサーバ

備考

デフォルトは 1460 バイトです。最小値は 300 バイトで、最大値は 16000 バイトです。

参照

- 「CommBufferSize 接続パラメータ [CBSIZE]」 234 ページ

-pc サーバ・オプション

同一コンピュータ接続以外のすべての接続のパケットを圧縮します。

構文

{ **dbsrv10** } **-pc** …

適用対象

すべてのオペレーティング・システムとネットワーク・サーバ (Web サーバを除く)

備考

SQL Anywhere のクライアントとサーバの間で送信されるパケットは、-pc オプションを使用して圧縮できます。状況によっては、接続を圧縮することでパフォーマンスが向上する場合があります。大幅に圧縮可能なデータの大規模なデータ転送では、圧縮率が高くなります。クライアントの接続パラメータに COMPRESS=NO を指定すると、特定のクライアントについてこのオプションを上書きできます。

デフォルトでは、接続は圧縮されません。-pc オプションを指定すると、同一コンピュータ接続、Web サービス接続、および TDS 接続を除くすべての接続のパケットが圧縮されます。TDS 接続 (jConnect を含む) では、SQL Anywhere 通信圧縮はサポートされません。

どの通信リンクを使用している場合でも、同一コンピュータ接続は -pc オプションまたは COMPRESS=YES 接続パラメータを使用しても圧縮されません。

参照

- 「パフォーマンス改善のための通信圧縮設定の調整」 130 ページ
- 「Compress 接続パラメータ [COMP]」 237 ページ
- 「圧縮機能の使用」 『SQL Anywhere サーバ - SQL の使用法』

-pt サーバ・オプション

パケットの圧縮が適用される最小パケット・サイズを増減します。

構文

{ **dbsrv10** } **-pt** *size* …

適用対象

すべてのオペレーティング・システムとネットワーク・サーバ

備考

このパラメータには、圧縮が適用されるパケット・サイズの最小バイト数を表す整数値を指定します。80 未満の値はおすすめしません。デフォルトは 120 バイトです。

状況によっては、圧縮のスレッショルドを変更すると、パケットの転送速度が上昇する場合のみパケットを圧縮できるようになり、圧縮された接続のパフォーマンスが向上することがあります。ほとんどの場合にデフォルト設定が適しています。

クライアントとサーバで圧縮スレッショルドの設定が異なる場合は、クライアントの設定が適用されます。

参照

- 「パフォーマンス改善のための通信圧縮設定の調整」 130 ページ
- 「CompressionThreshold 接続パラメータ [COMPTH]」 238 ページ
- 「圧縮機能の使用」 『SQL Anywhere サーバ - SQL の使用法』

-qi サーバ・オプション

データベース・サーバ・トレイ・アイコンとサーバ・メッセージ・ウインドウを表示するかどうかを制御します。

構文

{ dbsrv10 | dbeng10 } -qi …

適用対象

Windows

備考

このオプションは、起動エラー・ダイアログを除いて、サーバの実行中に視覚的な表示が出ないようにします。-o または -oe ログのいずれか (または両方) を使用してエラーを診断できます。

-qn サーバ・オプション

起動時にサーバ・メッセージ・ウィンドウを最小化しないことを指定します。

構文

{ dbsrv10 | dbeng10 } -qn …

適用対象

Windows

Linux (X-Window Server が使用されている場合)

備考

デフォルトでは、データベース・サーバの起動が完了すると、サーバ・メッセージ・ウィンドウは自動的に最小化されます。このオプションを指定すると、サーバ・メッセージ・ウィンドウはデータベース・サーバの起動後に最小化されません。

データベース・サーバを自動的に起動するアプリケーションがアクティブでなく、-qn が指定されている場合、サーバ・メッセージ・ウィンドウはバックグラウンドで表示されることがあります。

Linux では、-qn オプションとともに -ux オプション (X-Window Server の使用を指定) を指定する必要があります。

参照

- 「-ux サーバ・オプション」 203 ページ

- 「-qi サーバ・オプション」 187 ページ
- 「-qp サーバ・オプション」 188 ページ
- 「-qs サーバ・オプション」 188 ページ
- 「-qw サーバ・オプション」 189 ページ

例

次のコマンドは、Linux または Solaris 上でデータベース・サーバを起動し、サーバ・メッセージ・ウィンドウを表示して、データベース・サーバの起動が完了した後でサーバ・メッセージ・ウィンドウを最小化しません。

```
dbeng10 -ux -qn sample.db
```

-qp サーバ・オプション

サーバ・メッセージ・ウィンドウには表示されない、パフォーマンスに関するメッセージを指定します。

構文

{ dbsrv10 | dbeng10 } -qp …

適用対象

すべてのオペレーティング・システムとサーバ

備考

サーバ・メッセージ・ウィンドウにパフォーマンスのメッセージを表示しないようにします。表示されなくなるメッセージには次のものがあります。

- テーブル *table-name* のユニークでないインデックスまたはプライマリ・キー
- *nnn* フラグメントで構成される *mydatabase.db* データベース・ファイル

-qs サーバ・オプション

起動エラー・ダイアログを表示しないようにします。

構文

{ dbsrv10 | dbeng10 } -qs …

適用対象

Windows

備考

このオプションは、起動エラー・ダイアログを表示しないようにするものです。起動エラーには、次のようなエラーがあります。

- データベース・ファイル *database-file* が開けない／読み込めない

- その名前のデータベース・サーバがすでに起動している

Windows プラットフォームでは、サーバが自動的に起動しない場合、これらのエラーがダイアログに表示されるので、これらをクリアしてからサーバを停止する必要があります。これらのダイアログは -qs オプションを使用すると表示されません。

言語 DLL のロード時のエラーの場合、-qs がコマンド・ラインで指定されていても @data 構文に指定されていないとダイアログは表示されません。このエラーは -o または -oe で指定されたログには記録されず、Windows アプリケーション・イベント・ログに記録されます (Windows CE の場合を除く)。

-qs がコマンド・ラインで指定されていても、@data 拡張で指定されていない場合、使用法エラーは表示されません。

-qw サーバ・オプション

サーバ・メッセージ・ウィンドウを表示しないように指定します。

構文

{ dbsrv10 | dbeng10 } -qw …

適用対象

NetWare を除くすべてのオペレーティング・システムとサーバ

備考

このオプションを指定すると、サーバ・メッセージ・ウィンドウ (Windows プラットフォーム) とコンソール上のメッセージ (Windows 以外のプラットフォーム) が表示されなくなります。

-r サーバ・オプション

サーバ上で起動されるすべてのデータベースを、強制的に読み込み専用にします。データベースへの変更はできません。つまり、サーバはデータベース・ファイルやトランザクション・ログ・ファイルを変更しません。

構文

{ dbsrv10 | dbeng10 } -r …

適用対象

すべてのオペレーティング・システムとサーバ

備考

コマンド・ラインでどのデータベース名よりも前にオプションを指定した場合、テンポラリ・ファイルを除くすべてのデータベース・ファイルが読み込み専用モードで開きます。あるデータベース名の後ろに -r サーバ・オプションを指定した場合、そのデータベースだけが読み込み専用になります。テンポラリ・テーブルに変更を加えることはできますが、トランザクション・ロ

グとロールバック・ログが無効化されているため、ROLLBACK を実行しても効果はありません。

変更が不可能なデータベース・ファイルの例として、CD-ROM によって配賦されるデータベースがあります。このようなデータベースには読み込み専用モードでアクセスできます。

INSERT 文や DELETE 文などを使ってデータベースを変更しようとすると、SQLSTATE_READ_ONLY_DATABASE エラーが発生します。

リカバリを必要とするデータベースは、読み込み専用モードでは起動できません。たとえば、オンライン・バックアップを使って作成したデータベース・ファイルは、バックアップの開始時に開いているトランザクションがあると、読み込み専用モードで起動できません。これは、バックアップ・コピーの開始時に、これらのトランザクションがリカバリを必要とするためです。

監査がオンであるデータベースを読み込み専用モードで起動することはできません。

参照

- 「-r データベース・オプション」 224 ページ
- 「auditing オプション [データベース]」 428 ページ

例

2 つのデータベースを読み込み専用モードで開くには、次のように指定します。

```
dbeng10 -r database1.db database2.db
```

2 つのうち最初のデータベースだけを読み込み専用モードで開くには、次のように指定します。

```
dbeng10 database1.db -r database2.db
```

-s サーバ・オプション

syslog メッセージのユーザ ID を設定します。

構文

{ dbsrv10 | dbeng10 } -s { none | user | daemon | local*n* } …

適用対象

UNIX

備考

syslog 機能のメッセージで使用されるシステム・ユーザ ID を設定します。フォアグラウンドで起動しているデータベース・サーバのデフォルトは user で、バックグラウンドで起動しているサーバ (dbspawn で起動した場合、クライアントが自動的に起動した場合、-ud データベース・サーバ・オプションで起動した場合など) のデフォルトは daemon です。

none を指定すると、syslog メッセージはログに記録されません。引数 local*n* を使用すると、機能識別子を使用してメッセージをファイルへリダイレクトできます。*n* に 0 ー 7 の数字を指定できます。詳細については、UNIX syslog(3) man ページを参照してください。

次の手順は Solaris でのメッセージのリダイレクト方法を説明したものですが、Linux、AIX、Mac OS X でも使用できます。HP-UX などのプラットフォームでは、*syslog.conf* ファイルは別の場所にあります。*/var/adm/sqlanywhere* ファイルは、任意の場所に配置できます。

♦ **機能識別子を使用してメッセージをファイルへリダイレクトするには、次の手順に従います。**

1. システムで実行している他のアプリケーションが使用していないユニークな機能識別子を選択します。

   /etc/syslog.conf ファイルを見ると、local*n* 機能が参照する識別子を確認できます。

2. */etc/syslog.conf* ファイルを開き、次の行を追加します。local*n* は手順 1 で選択した機能識別子です。

   ```
   localn.err;localn.info;localn.notice  /var/adm/sqlanywhere
   ```

3. */var/adm/sqlanywhere* ファイルを作成します。

   ```
   touch /var/adm/sqlanywhere
   ```

4. syslogd のプロセス ID を検索して、*syslog.conf* ファイルを変更したことを syslogd プロセスに通知します。

   ```
   ps -ef | grep syslogd
   ```

   次に、以下のコマンドを実行します。*pid* は syslogd のプロセス ID です。

   ```
   kill -HUP pid
   ```

5. 次のコマンドを使用して SQL Anywhere データベース・サーバを起動します。local*n* は手順 1 で選択した機能識別子です。

   ```
   dbeng10 -s localn ...
   ```

   これで、SQL Anywhere データベース・サーバが syslog にレポートするメッセージは */var/adm/sqlanywhere* ファイルにリダイレクトされます。

-sb サーバ・オプション

ブロードキャストに対するサーバの動作を指定します。

構文

{ dbsrv10 | dbeng10 } -sb { 0 | 1 } …

適用対象

SPX、TCP/IP

備考

-sb 0 を使用すると、サーバで UDP または IPX ブロードキャスト・リスナが起動されなくなります。このオプションを指定すると、クライアントがサーバに接続するときに DoBroadcast=NONE オプションと HOST= オプションを使用するとともに、dblocate の使用時にはサーバがリストから除外されます。

-sb 1 を使用すると、サーバが dblocate からのブロードキャストに応答しなくなります。これは接続論理には影響しません。LINKS=tcpip と ENG=*name* を指定することにより、サーバに接続できます。

-sf サーバ・オプション

このデータベース・サーバで実行されるデータベースの機能を保護します。

構文

{ **dbsrv10** | **dbeng10** } **-sf** *feature-list* …

適用対象

すべてのオペレーティング・システムとサーバ

備考

このオプションでは、データベース・サーバの機能を有効または無効にできます。この設定は、データベース・サーバ上のすべてのデータベースに影響します。-sk オプションで指定されたキーを secure_feature_key オプションに設定することによって、無効になっている (保護されている) すべての接続機能を有効化できます。secure_feature_key オプションが -sk で指定されたキーに設定された接続では、sa_server_option システム・プロシージャを使用して、データベース・サーバで保護されている機能セットを変更することもできます。

feature-list は、データベース・サーバで保護する機能名または機能セットをカンマで区切って示したリストです。*feature-name* は機能を無効にすることを示し、*-feature-name* は機能を無効化された機能のリストから削除することを示します。たとえば、次のコマンドは、DB 領域機能だけを有効にします。

```
dbeng10 -n secure_server -sf all,-dbspace
```

次の *feature-name* 値がサポートされています (カッコ内の値は機能名の省略形で、機能名と同じように指定できます)。

機能名	説明
backup	BACKUP 文を使用不可にします。これにより、サーバ側のバックアップを実行できなくなります。その場合でも、クライアント側のバックアップは dbbackup を使用して実行できます。「BACKUP 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
database	CREATE DATABASE 文、ALTER DATABASE 文、DROP DATABASE 文、CREATE ENCRYPTED FILE 文、CREATE DECRYPTED FILE 文を使用不可にします。
db_delete_file (delete_file)	データベース・ファイルを削除する db_delete_file DBLib 関数を使用不可にします。db_delete_file は dbbackup -x オプションと -xo オプションで使用されるため、db_delete_file を保護すると、-x または -xo オプションが指定された dbbackup は失敗します。「db_delete_file 関数」『SQL Anywhere サーバ - プログラミング』を参照してください。

機能名	説明
dbspace	CREATE DBSPACE 文、ALTER DBSPACE 文、DROP DBSPACE 文を使用不可にします。
directory (dir)	ディレクトリ・クラス・プロキシ・テーブルを使用不可にします。この機能は、remote_data_access をオフに設定した場合にも無効になります。
external_procedure (ext_proc)	外部ストアド・プロシージャを使用不可にします。これによってデータベース・サーバに組み込まれている xp_* システム・プロシージャ (xp_cmdshell や xp_readfile など) が*使用不可になることはありません*。「プロシージャからの外部ライブラリの呼び出し」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
java	Java 関連の機能 (Java プロシージャなど) を無効にします。「チュートリアル：データベースにおける Java の使用」『SQL Anywhere サーバ - プログラミング』を参照してください。
load_table (load)	LOAD TABLE 文を使用不可にします。「LOAD TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
log_file (log)	ログ・ファイルの変更機能とログ・ファイルの最大サイズを拡大する機能を無効にします。データベース・サーバの起動時には、サーバ・ログ・ファイルとそのサイズを指定できます。
remote_data_access (proxy)	プロキシ・テーブルなどのリモート・データ・アクセス・サービスを使用不可にします。
request_log (rll)	要求ログ・ファイルの変更機能とその最大サイズと最大ファイル数を増加する機能を無効にします。データベース・サーバの起動コマンドには、要求ログ・ファイルとその最大サイズを指定できます。ただし、それらをサーバの起動後に変更することはできません。要求ログ機能を無効にしても、要求ロギングのオンとオフを切り替えたり、要求ログ・ファイルの最大サイズや最大数を小さくしたりすることは可能です。「要求ロギング」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
restore	RESTORE DATABASE 文を使用不可にします。「RESTORE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
unload	UNLOAD TABLE 文と UNLOAD 文を使用不可にします。「UNLOAD TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』と「UNLOAD 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
web_service_client (web_client)	外部 HTTP または SOAP Web サービスに対するリモート・コールとして定義されたストアド・プロシージャを使用不可にします。

機能名	説明
xp_cmdshell (cmdshell)	xp_cmdshell プロシージャを使用不可にします。「xp_cmdshell システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
xp_read_file (read_file)	xp_read_file プロシージャを使用不可にします。「xp_read_file システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
xp_write_file (write_file)	xp_write_file プロシージャを使用不可にします。「xp_write_file システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

以下の機能セットを指定すると、関連する複数の機能を無効にできます。次の値がサポートされます。

機能セット	説明
all	無効にすることが可能なすべての機能 (上記のリスト内の機能) を無効にします。
local_call	サーバが所有および制御していないコードの実行を可能にするすべての機能を無効にします。この機能セットには、cmdshell、external_procedure、java が属します。
local_db	データベース・ファイル関連のすべての機能を無効にします。この機能セットには、backup、restore、database、dbspace が属します。
local_io	ファイルとその内容への直接アクセスを可能にするすべての機能を無効にします。この機能セットには、db_delete_file、xp_read_file、xp_write_file、directory、load_table、unload が属します。
local_log	ディスク上にファイルを作成し、データを直接書き込む、すべてのロギング機能を無効にします。この機能セットには、request_log と log_file が属します。
local	すべての local 機能を無効にします。この機能セットには、local_call、local_db、local_io、local_log が属します。
none	いずれの機能も無効にしないことを指定します。
remote	リモート・アクセスまたはリモート・プロセスとの通信を可能にするすべての機能を無効にします。この機能セットには、web_service_client と remote_data_access が属します。

参照

- 「-sk サーバ・オプション」 195 ページ
- 「secure_feature_key [データベース]」 494 ページ
- 「sa_server_option システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』
- 「保護された機能の指定」 928 ページ

例

次のコマンドは、要求ログとすべてのリモート・データ・アクセス機能へのアクセスを無効にしてデータベース・サーバ secure_server を起動します。これらの機能は、-sk で指定したキーを secure_feature_key データベース・オプションに設定して、特定の接続において有効にできます。

```
dbsrv10 -n secure_server -sf request_log,remote -sk j978kls12
```

secure_server データベース・サーバ上のデータベースに接続するときに secure_feature_key オプションを -sk で指定された値に設定すると、その接続において、要求ログへのアクセスとリモート・データ・アクセス機能が有効になります。

```
SET TEMPORARY OPTION secure_feature_key = 'j978kls12';
```

次のコマンドは、ローカル・データベース機能を除き、すべての機能を無効にします。

```
dbeng10 -n secure_server -sf all,-local_db
```

-sk サーバ・オプション

データベース・サーバで無効になっている機能を有効にするためのキーを指定します。

構文

{ **dbsrv10** | **dbeng10** } **-sk** *key* ...

適用対象

すべてのオペレーティング・システムとサーバ

備考

-sf オプションを使用してデータベース・サーバの機能を保護するときには、-sk オプションを使用して、機能を有効にするキーも指定できます。このキーを secure_feature_key データベース・オプションに指定すると、保護されている機能が有効になります。この場合、sa_server_option システム・プロシージャを使用して、データベース・サーバ上で実行されているすべてのデータベースを保護する機能または機能セットに変更を加えることもできます。

secure_feature_key オプションを -sk で指定された値以外の値に設定すると、エラーは発生せず、-sf で指定された機能はその接続において保護されたままとなります。

参照

- 「-sf サーバ・オプション」 192 ページ
- 「secure_feature_key [データベース]」 494 ページ
- 「sa_server_option システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
- 「保護された機能の指定」 928 ページ

例

次のコマンドは、バックアップ機能へのアクセスを無効にしてデータベース・サーバ secure_server を起動します。これらの機能は、-sk で指定したキーを後で使用して、特定の接続において有効にできます。

```
dbsrv10 -n secure_server -sf backup -sk j978kls12
```


secure_server データベース・サーバ上のデータベースに接続するときに secure_feature_key オプションを -sk で指定された値に設定すると、バックアップの実行が可能となり、secure_server データベース・サーバ上で無効になっている機能を変更できます。

```
SET TEMPORARY OPTION secure_feature_key = 'j978kls12';
```

その後、次のコマンドを実行して、secure_server データベース・サーバ上で実行されているデータベースに対して、保護されたすべての機能を無効にできます。

```
CALL sa_server_option( 'SecureFeatures', 'all' )
```

-su サーバ・オプション

ユーティリティ・データベース (utility_db) の DBA ユーザのパスワードを設定します。または、ユーティリティ・データベースへの接続を無効にします。

構文

{ **dbsrv10** | **dbeng10** } **-su** *password* …

適用対象

すべてのオペレーティング・システムとサーバ

備考

このオプションは、ユーティリティ・データベースの DBA ユーザの最初のパスワードを指定します。パスワードでは大文字と小文字が区別されます。ユーティリティ・データベースへのすべての接続を無効にするには、パスワードに **none** を指定します。ユーティリティ・データベースのパスワードをコマンド・ライン上でクリア・テキストで入力することを回避するには、dbfhide を使用してパスワードを含むファイルの内容を読みにくくし、難読化されたファイルをコマンド・ライン上で参照します。

パーソナル・データベース・サーバを使用している場合、-su オプションを指定しないと、DBA ユーザ ID と任意のパスワードを使用してユーティリティ・データベースに接続できます。ネットワーク・データベース・サーバを使用している場合で、-su オプションを指定しないときに、ユーティリティ・データベースに接続するには、*util_db.ini* ファイルが存在することと、DBA ユーザ ID および *util_db.ini* ファイルの内容と一致するパスワードを使用することが必要です。ネットワーク・サーバで -su と *util_db.ini* ファイルの両方を使用すると、*util_db.ini* ファイルが無視されます。*util_db.ini* ファイルの使用は推奨されません。

utility_db に接続しているときに、GRANT CONNECT TO DBA IDENTIFIED BY '*new-password*' 文を実行して、ユーティリティ・データベースの DBA ユーザのパスワードを変更できます。utility_db データベースへの接続を無効にするには、REVOKE CONNECT FROM DBA 文を使用します。

参照

- 「ファイル非表示ユーティリティ (dbfhide)」 657 ページ
- 「GRANT 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「REVOKE 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「ユーティリティ・データベースへの接続」 299 ページ

例

次のコマンドは、ユーティリティ・データベースへのすべての接続を無効にします。

```
dbeng10 -su none c:¥inventory.db
```

次の例では、dbfhide を使用して、ユーティリティ・データベースのパスワードを含む *util_db_pwd.cfg* ファイルを読みにくくし、ファイル名を *util_db_pwd_hide.cfg* に変更します。

```
dbfhide util_db_pwd.cfg util_db_pwd_hide.cfg
```

その後、*util_db_pwd_hide.cfg* ファイルを使用して、ユーティリティ・データベースのパスワードを指定できます。

```
dbsrv10 -su @util_db_pwd_hide.cfg -n my_server c:¥inventory.db
```

-ti サーバ・オプション

非アクティブ接続を切断します。

構文

{ **dbsrv10** | **dbeng10** } **-ti** *minutes* …

適用対象

すべてのオペレーティング・システムとサーバ

備考

minutes で指定された時間の間、要求を送信しなかった接続を切断します。デフォルト値は 240 (4 時間) です。最大値は 32767 です。データベース・トランザクション中のクライアント・コンピュータは、トランザクションが終了するか、接続が終了するまでロックされます。-ti オプションを指定すると、非アクティブ接続が切断され、ロックが解除されます。

ほとんどの接続はネットワーク・リンク (TCP または SPX) で行われているため、-ti オプションは dbsrv10 と一緒に使用すると非常に便利です。

-ti オプションは、ローカル TCP/IP 接続の場合のみ dbeng10 と 一緒に使用すると便利です。-ti を使用しても、共有メモリを使用しているローカル・サーバへの接続には影響しません。

値を 0 に設定すると、非アクティブ接続は検査されず、接続が切断されません。

参照

- 「sa_server_option システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』

-tl サーバ・オプション

活性パケットを送信する期間を設定します。

構文

{ **dbsrv10** | **dbeng10** } **-tl** *seconds* …

適用対象

TCP/IP または SPX を使用するすべてのデータベース・サーバ

備考

接続が維持されていることを確認するため、クライアント／サーバの TCP/IP または SPX 通信プロトコルを介して、定期的に活性パケットが送信されます。接続で活性パケットを検出することなく、指定した LivenessTimeout 時間 (デフォルトは 2 分) にわたってサーバが実行されていると、通信は切断され、サーバはそのクライアントに関連付けられている接続を削除します。非スレッドの UNIX クライアントと TDS 接続では、活性パケットによる確認は行われません。

サーバで -tl オプションを指定すると、活性期間が指定されていないすべてのクライアントに対して LivenessTimeout 値を設定できます。

LivenessTimeout 値の 3 分の 1 から 3 分の 2 の期間で接続がパケットを送信しない場合に、活性パケットが送信されます。

200 以上の接続がある場合に、サーバは指定された LivenessTimeout 値に基づいて LivenessTimeout 値が高いものを自動的に算出します。これにより、サーバはより多くの接続を効率よく処理できます。活性パケットは、各アイドル接続において、LivenessTimeout 値の 3 分の 1 から 3 分の 2 の期間で送信されます。大量の活性パケットが同時に送信されることはありません。(ネットワーク、コンピュータのハードウェア、コンピュータの CPU とネットワーク負荷などの影響で) 活性パケットの送信に時間がかかる場合、LivenessTimeout 値の 3 分の 2 の期間が経過した後で活性パケットを送信することもできます。活性パケットの送信に時間がかかる場合、サーバ・コンソールに警告が表示されます。この警告が発生したら、LivenessTimeout 値の増加を検討してください。

これは一般的にはおすすめしませんが、次のように指定して活性タイムアウトを無効にできます。

```
dbsrv10 -tl 0
```

LivenessTimeout オプションを無効にせずに、次のように値を 1 時間に増やすことを検討してください。

```
dbsrv10 -tl 3600
```

参照

- 「sa_server_option システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』

-tmf サーバ・オプション

異常な状態での、分散トランザクションのリカバリに使用します。

構文

{ **dbsrv10** | **dbeng10** } **-tmf** …

適用対象

Windows

備考

このオプションは、分散トランザクション・コーディネータが使用できない場合に、分散トランザクションのリカバリ中に使用します。また、分散トランザクション・コーディネータが使用できないプラットフォームで、トランザクション・ログに分散トランザクションがあるデータベースを起動する場合にも使用できます。

警告
このオプションを使用すると、分散トランザクションは正常にはリカバリされません。通常の作業では使用しないでください。

参照

- 「分散トランザクションからのリカバリ」『SQL Anywhere サーバ - プログラミング』

-tmt サーバ・オプション

分散トランザクションに参加するための再エンリスト・タイムアウトを設定します。

構文

{ **dbsrv10** | **dbeng10** } **-tmt** *milliseconds* …

適用対象

Windows

備考

分散トランザクションのリカバリ時に使用します。この値は、データベース・サーバが再登録されるまでの待機時間を指定します。デフォルトでは、タイムアウトはありません (データベース・サーバは、無期限に待機します)。

参照

- 「分散トランザクションからのリカバリ」『SQL Anywhere サーバ - プログラミング』

-tq サーバ・オプション

指定の時刻にサーバを停止します。

構文

{ **dbsrv10** | **dbeng10** } **-tq** { *datetime* | *time* } …

適用対象

すべてのオペレーティング・システムとサーバ

備考

このオプションは、自動オフライン・バックアップ・プロシージャの設定に役立ちます。「バックアップとデータ・リカバリ」 811 ページを参照してください。

時間のフォーマットは *hh*:*mm* (24 時間表記) で、オプションで前に日付を付けることができます。日付を指定する場合は、日付と時間を二重引用符で囲んで、"*YYYY/MM/DD HH*:*MM*" の形式にする必要があります。

参照

- 「sa_server_option システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』

-u サーバ・オプション

オペレーティング・システムのディスク・キャッシュを使用してファイルを開きます。

構文

{ **dbsrv10** | **dbeng10** } **-u** ⋯

適用対象

Windows、UNIX

備考

データベース・キャッシュに加え、オペレーティング・システムのディスク・キャッシュも使ってファイルが開かれます。

場合によっては、オペレーティング・システムのディスク・キャッシュを使用することでパフォーマンスが向上しますが、このオプションを使わずにデータベース・キャッシュだけを使っても、通常は十分なパフォーマンスを得ることができます。

サーバを専用コンピュータで実行している場合は、-u オプションを使用しないでください。通常、データベース・キャッシュを使用する方が効率的です。-u オプションを使用するのは、サーバを実行するコンピュータに他にもいくつかのアプリケーションがインストールされていて (サイズの大きなデータベース・キャッシュが他のアプリケーションを実行するための障害となり)、しかも I/O の多いタスクをサーバ上で断続的に実行する (キャッシュ・サイズを大きくすることでパフォーマンスが向上する) 場合です。

-ua サーバ・オプション

非同期 I/O の使用をオフにします。

構文

{ **dbsrv10** | **dbeng10** } **-ua** ...

適用対象

Linux

備考

デフォルトで、使用可能な場合にデータベース・サーバは Linux で非同期 I/O を使用します。非同期 I/O を使用するには、以下の条件を満たす必要があります。

1. ライブラリ *libaio.so* が実行時にロードできる。
2. カーネルが非同期 I/O をサポートしている。

非同期 I/O の使用をオフにしたい場合、データベース・サーバ・コマンド・ラインで -ua オプションを指定します。

-uc サーバ・オプション

データベース・サーバをコンソール・モードで起動します。これはデフォルトです。

構文

{ dbsrv10 | dbeng10 } -uc

適用対象

UNIX

備考

データベース・サーバをコンソール・モードで起動します。-uc、-ui、-ux のうち 1 つだけを指定してください。-uc を指定すると、データベース・サーバはソフトウェアの以前のリリースと同じ方法で起動されます。

デーモンとしてのデータベース・サーバの起動の詳細については、「-ud サーバ・オプション」 201 ページを参照してください。

参照

- 「-ui サーバ・オプション」 202 ページ
- 「-ux サーバ・オプション」 203 ページ

-ud サーバ・オプション

デーモンとして実行します。

構文

{ dbsrv10 | dbeng10 } -ud …

適用対象

UNIX

備考

このオプションを使用すると、現在のオペレーティング・システム・セッションが終了しても引き続きサーバを実行できます。

-ud オプションを使用してデーモンを直接起動したときは、dbeng10 コマンドと dbsrv10 コマンドがデーモン・プロセスを作成し、(終了して次のコマンドを実行できるように) すぐに返します。その後、デーモンがそれ自体を初期化するか、コマンドで指定されたデータベースを開こうとします。

-ud オプションの代わりに dbspawn を使用することの利点は、デーモンが起動し、要求を受け入れる状態になったことを確認するまで dbspawn プロセスが終了しないことです。何らかの理由でデーモンの起動が失敗した場合、dbspawn の終了コードは 0 以外の値になります。

参照

- 「サーバ・バックグラウンド起動ユーティリティ (dbspawn)」 730 ページ
- 「ソフトウェア・コンポーネントの終了コード」 『SQL Anywhere サーバ - プログラミング』

-uf サーバ・オプション

致命的なエラーの発生時に実行するアクションを指定します。

構文

{ dbsrv10 | dbeng10 } -uf *action* …

適用対象

UNIX

備考

このオプションでは、致命的なエラーが発生したときに以下のアクションのうちどれを実行するかを指定します。

- **abort** UNIX の abort 関数が呼び出され、コア・ファイルが生成されます。
- **default** データベース・サーバは、いずれの場合も abort と同じ動作をします。ただし、デバイス・フルの致命的なエラーが発生した場合は、defunct と同じ動作となります。これによって、システムはフル・デバイスとなってもコア・ファイルを作成しようとしません。これはデフォルトの動作です。
- **defunct** データベース・サーバは実行を続け、abort を呼び出しません。データベース・サーバに接続しようとすると、元の致命的なエラーの SQL エラーを受け取ります。

参照

- 「-ux サーバ・オプション」 203 ページ

-ui サーバ・オプション

[サーバ起動オプション] ダイアログを開いてサーバ・メッセージ・ウィンドウを表示します。また、Linux で使用可能な表示がない場合は、コンソール・モードでデータベース・サーバを起動します (X-Window Server が起動するかどうかにかかわらず、データベース・サーバを起動します)。

構文

{ dbsrv10 | dbeng10 } -ui

適用対象

X-Window Server がサポートされている Linux

備考

-ui オプションを使用すると、[サーバ起動オプション] ダイアログを使用して、データベース・サーバ起動時のサーバ・オプションを指定し、サーバが起動したらサーバ・メッセージ・ウィンドウを表示できます。

-ui を指定すると、サーバは使用可能な表示を探そうとします。たとえば、DISPLAY 環境変数が設定されていなかったり、X-Window Server が実行されていなかったりしたために、使用可能な表示が見つからなかった場合は、データベース・サーバはコンソール・モードで起動されます。使用可能な表示が見つからない場合にデータベース・サーバの起動を中止するには、-ui オプションではなく -ux オプションを指定します。-uc、-ui、-ux のうち 1 つだけを指定してください。

サーバ・コマンド・ラインで -ux オプションのみを指定した場合は、[サーバ起動オプション] ダイアログが表示されるので、データベース・サーバを起動するためのオプションを入力できます。サーバが起動されると、サーバ・メッセージのダイアログが表示されます。-ux の他にもサーバ・オプションを指定した場合は、データベース・サーバを起動するとサーバ・メッセージ・ウィンドウが表示されます。

デーモンとしてのデータベース・サーバの起動の詳細については、「-ud サーバ・オプション」 201 ページを参照してください。

参照

- 「-uc サーバ・オプション」 201 ページ
- 「-ux サーバ・オプション」 203 ページ

-ut サーバ・オプション

テンポラリ・ファイルをタッチします。

構文

{ **dbsrv10** | **dbeng10** } **-ut** *minutes* …

適用対象

UNIX

備考

このオプションを使用すると、指定の間隔でサーバにテンポラリ・ファイルをタッチさせることができます。

-ux サーバ・オプション

[サーバ起動オプション] ダイアログを開くか、サーバ・メッセージ・ウィンドウ (Linux の場合) を表示します (X-Window Server を使用します)。

構文

{ **dbsrv10** | **dbeng10** } **-ux**

適用対象

X-Window Server がサポートされている Linux

備考

-ux オプションを使用すると、データベース・サーバの起動時に 2 つの操作が行えます。それは、[サーバ起動オプション] ダイアログを使用してデータベース・サーバ起動時にサーバ・オプションを指定することと、サーバが起動した後にサーバ・メッセージ・ウィンドウを表示することです。

サーバ・コマンド・ラインで -ux オプションのみを指定した場合は、[サーバ起動オプション] ダイアログが表示されるので、データベース・サーバ起動のオプションを入力できます。

-ux が指定されている場合、サーバは使用可能な表示を見つけます。たとえば、DISPLAY 環境変数が設定されていなかったり、X-Window Server が実行されていなかったりしたために、使用可能な表示が見つからなかった場合、データベース・サーバは起動できません。使用可能な表示が見つからない場合でもデータベース・サーバを起動するには、-ux の代わりに -ui オプションを使用します。

-ux の他にもサーバ・オプションを指定した場合は、データベース・サーバが起動するとサーバ・メッセージ・ウィンドウが表示されます。-uc、-ui、-ux のうち 1 つだけを指定してください。

デーモンとしてのデータベース・サーバの起動の詳細については、「-ud サーバ・オプション」 201 ページを参照してください。

参照

- 「-uc サーバ・オプション」 201 ページ
- 「-ui サーバ・オプション」 202 ページ
- 「-qn サーバ・オプション」 187 ページ

例

次のコマンドを入力すると、データベース・サーバの起動のオプションを入力するための [サーバ起動オプション] ダイアログが表示されます。

```
dbeng10 -ux
```

次のコマンドを入力すると、データベース・サーバが起動され、サーバ・メッセージ・ウィンドウが表示されます。

```
dbeng10 -ux sample.db
```

-v サーバ・オプション

ソフトウェアのバージョンを表示します。

構文

{ **dbsrv10** | **dbeng10** } **-v** …

適用対象

すべてのオペレーティング・システムとサーバ

備考

メッセージ・ボックスにデータベース・サーバのバージョンを表示して、停止します。

-x サーバ・オプション

サーバ側のネットワーク通信プロトコルを指定します。

構文 1

dbsrv10 -x { **all** | **none** | *srv-protocols* } …

srv-protocols:
{ [**spx** | **tcpip**] *parmlist* },…
parmlist:
(*parm=value*;…**)**

構文 2

dbeng10 -x { **all** | **none** | *eng-protocols* } …

eng-protocols:
{ **tcpip** [*parmlist*] },…
parmlist:
(*parm=value*;…**)**

適用対象

すべてのオペレーティング・システムとサーバ

備考

-x オプションを使用して、クライアント接続ブロードキャストの受信に使用する通信プロトコル (および共有メモリ) を指定します。

-x オプションを指定しないと、サーバは、共有メモリ・プロトコルも含め、オペレーティング・システムで実行しているデータベース・サーバでサポートされるすべてのプロトコルを使用して、クライアント接続ブロードキャストを受信しようとします。

-x オプションとともに 1 つ以上のプロトコルを指定すると、サーバは、指定したプロトコルと共有メモリ・プロトコルを使用して、クライアント接続ブロードキャストを受信しようとします。

UNIX の共有メモリ接続の保護の詳細については、「セキュリティのヒント」 920 ページを参照してください。

注意
Windows CE を実行しているサーバで -x オプションを指定すると、特に明示的に指定しないかぎり、クライアント接続ブロードキャストの受信に TCP/IP プロトコルのみを使用しようとします。

-x オプションについて選択した設定に関係なく、サーバは常に共有メモリ・プロトコルを使用して接続ブロードキャストを受信します。共有メモリ・プロトコル以外に、次のプロトコルを指定できます。

- **ALL** 共有メモリ・プロトコルを含め、このプラットフォーム上のサーバでサポートされているすべての通信プロトコルを使用するクライアントによる接続試行を受信します。これはデフォルトです。
- **NONE** 共有メモリ・プロトコルのみを使用するクライアントによる接続試行を受信します。
- **SPX** SPX プロトコルを使用するクライアントによる接続試行を受信します。SPX プロトコルは NetWare と Windows のネットワーク・サーバでサポートされています。
- **TCPIP (TCP)** TCP/IP プロトコルを使用してクライアントに接続しようとします。TCP/IP プロトコルは、すべてのオペレーティング・システム上のネットワーク・サーバ、同一コンピュータ通信用のパーソナル・データベース・サーバでサポートされています。
  デフォルトでは、データベース・サーバはポート 2638 でブロードキャストを受信し、適切なポートにリダイレクトします。これで、ほとんどの場合に接続が保証されます。オプション -sb 0 を設定するか、BroadcastListener オプションをオフ (BroadcastListener=0) にすることで、このデフォルトを上書きし、サーバがポート 2638 で受信しないようにすることもできます。さらに、クライアントとサーバがファイアウォールを介して通信している場合、クライアントは、DoBroadcast=None と Host= を指定して、サーバが受信している正確なポートにパケットを送信する必要があります。

「ServerPort プロトコル・オプション [PORT]」 283 ページを参照してください。

プロトコルによっては、次のフォーマットでパラメータを追加できます。

```
-x tcpip(PARM1=value1;PARM2=value2;...)
```

使用可能なパラメータの詳細については、「ネットワーク・プロトコル・オプション」 265 ページを参照してください。

UNIX では、複数のパラメータを指定する場合に二重引用符が必要です。

```
-x "tcpip(PARM1=value1;PARM2=value2;...)"
```

参照

- 「CommLinks 接続パラメータ [LINKS]」 235 ページ

例

次の場合は、共有メモリと TCP/IP 通信だけが許可されます。

```
-x tcpip
```


-xa サーバ・オプション

監視サーバに対するデータベース名と認証文字列のカンマ区切りのリストを指定します。

構文

-xa auth=*auth-strings*;**DBN=***database-names*

適用対象

すべてのオペレーティング・システム、ネットワーク・サーバのみ

備考

このオプションは、データベース・ミラーリング・システムで監視サーバを起動する場合のみ指定します。

指定する認証文字列は、プライマリ・サーバとミラー・サーバで指定された認証文字列と一致している必要があります。

認証文字列のリストとデータベース名のリストにどちらもエントリが 1 つしかない場合、サーバは 1 つのデータベース・ミラーリング・システムの監視サーバとして動作します。それ以外の場合、2 つのリストのエントリ数は同じである必要があります。

参照

- 「DatabaseName 接続パラメータ [DBN]」 242 ページ
- 「-sn オプション」 225 ページ
- 「-xf サーバ・オプション」 207 ページ
- 「-xp データベース・オプション」 227 ページ

例

次のコマンドは、arbiter という監視サーバを起動します。

```
dbsrv10 -x tcpip -n arbiter -xa AUTH=abc;DBN=demo -xf c:¥arbiterstate.txt
```

-xf サーバ・オプション

データベース・ミラーリング・システムに関するステータス情報の管理に使用されるファイルのロケーションを指定します。

構文

-xf *state-file* …

適用対象

すべてのオペレーティング・システム、ネットワーク・サーバのみ

備考

-xf オプションでは、データベース・ミラーリング・システムに関するステータス情報の管理に使用されるファイルのロケーションを指定します。データベース・ミラーリングには、このオプションは必須です。デフォルトでは、ステータス情報ファイルは *server-name.mirror_state* という名前です。

データベース・ミラーリングに関するステータス情報の詳細については、「ステータス情報ファイル」 895 ページを参照してください。

参照

- 「-sn オプション」 225 ページ
- 「-xa サーバ・オプション」 207 ページ
- 「-xp データベース・オプション」 227 ページ

例

次のコマンドは (全体を 1 行に入力)、ステータス情報ファイル *c:¥server1state.txt* を使用するデータベース・サーバ server1 を起動します。

```
dbsrv10.exe -n server1 -x tcpip{DOBROADCAST=no}
-xf c:¥server1state.txt mydemo.db -sn mirrordemo
-xp partner=(ENG=server2;
AUTH=abc;ARBITER=(ENG=arbiter;LINKS=tcpip(TIMEOUT=1));
MODE=sync;LINKS=tcpip(TIMEOUT=1))
```

-xs サーバ・オプション

サーバ側の Web サービス通信プロトコルを指定します。

構文

{ dbeng10 | dbsrv10 } -xs { NONE | *protocol* **}** …

protocol : { **HTTP [(** *option=value*;…**)]**
| HTTPS [([FIPS={ Y | N };]*option=value*;…**)]**
[CERTIFICATE=*server-identity-filename*;
CERTIFICATE_PASSWORD=*password* **)] }** , ...

適用対象

すべてのオペレーティング・システムとサーバ

備考

-xs オプションを使用して、要求の受信に使用する Web プロトコルを指定します。

-xs オプションを指定しない場合、データベース・サーバは Web 要求を受信しようとしません。

-xs オプションとともに 1 つ以上のプロトコルを指定すると、サーバは、指定したプロトコルを使用して、Web 要求を受信しようとします。

> **注意**
> 複数の Web サーバを同時に起動する場合、どちらも同じデフォルト・ポートを使用するため、どちらか 1 つのポートを変更する必要があります。

トランスポート・レイヤ・セキュリティには、HTTPS または FIPS 承認の HTTPS プロトコルを使用できます。「SQL Anywhere Web サービスでのトランスポート・レイヤ・セキュリティの使用」 966 ページを参照してください。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」『SQL Anywhere 10 - 紹介』を参照してください。

-xs オプションで指定した設定に関係なく、サーバは常に共有メモリ・プロトコルを使用して接続試行を受信します。次のいずれかを指定できます。

- ***option*** 各プロトコルでサポートされている *option* の値のリストについては、「ネットワーク・プロトコル・オプション」 265 ページを参照してください。
- **HTTP** HTTP プロトコルを使用するクライアントによる Web 要求を受信します。受信するデフォルトのポートは 80 です。
- **HTTPS** HTTPS プロトコルを使用するクライアントによる Web 要求を受信します。受信するデフォルトのポートは 443 です。HTTPS を使用するためには、サーバの証明書とパスワードを指定する必要があります。HTTPS は RSA 暗号化を使用するため、パスワードは RSA 証明書であることが必要です。

  HTTPS のみ、または FIPS 承認の RSA 暗号化の場合は **FIPS=Y** を付けて **HTTPS** を指定できます。FIPS 承認の HTTPS は承認された別個のライブラリを使用しますが、HTTPS と互換性があります。

  注意
  FIPS 承認の HTTPS を使用する場合は、Mozilla Firefox ブラウザに接続できます。ただし、FIPS 承認の HTTPS が使用する暗号化パッケージ・プログラムは、 Internet Explorer、Opera、または Safari ブラウザではサポートされていません。FIPS 承認の HTTPS を使用する場合、これらのブラウザでは接続できません。

  FIPS 承認のアルゴリズムの実行については、「-fips サーバ・オプション」 166 ページを参照してください。

  - ***server-identity-filename*** サーバ ID のパスとファイル名を指定します。HTTPS では、RSA 証明書を使用する必要があります。
  - ***password*** サーバのプライベート・キーのパスワードを指定します。このパスワードは、サーバ証明書を作成するときに指定します。
- **NONE** Web 要求を受信しません。これはデフォルトです。

使用可能なパラメータの詳細については、「ネットワーク・プロトコル・オプション」 265 ページを参照してください。

UNIX では、複数のパラメータを指定する場合に二重引用符が必要です。

```
-xs "HTTP(OPTION1=value1;OPTION2=value2;...)"
```

例

HTTP Web 要求をポート 80 で受信します。


```
dbeng10 web.db -xs HTTP(PORT=80)
```

HTTPS を使用して Web 要求を受信します。

```
dbeng10 web.db -xs HTTPS
(FIPS=N;PORT=82;CERTIFICATE=sample.crt;CERTIFICATE_PASSWORD=tJ1#m6+W)
```

-z サーバ・オプション

診断通信メッセージなどのメッセージをトラブルシューティングのために表示します。

構文

{ **dbsrv10** | **dbeng10** } **-z** …

適用対象

すべてのオペレーティング・システムとサーバ

備考

このオプションは、問題の原因を突き止める場合にだけ使用します。情報は、サーバ・メッセージ・ウィンドウに表示されます。

-ze オプション

データベース・サーバ環境変数をサーバ・メッセージ・ウィンドウに表示します。

構文

{ **dbsrv10** | **dbeng10** } **-ze** …

適用対象

NetWare と Windows CE を除くすべてのオペレーティング・システムとサーバ

備考

-ze オプションを指定すると、起動時に環境変数がサーバ・メッセージ・ウィンドウに表示されます。サーバ・メッセージ・ウィンドウの内容をファイルに保存するには、データベース・サーバの起動時に -o オプションを指定します。

参照

- 「SQL Anywhere の環境変数」 305 ページ
- 「-o サーバ・オプション」 182 ページ

例

次のコマンドは、データベース・サーバ myserver を起動し、このサーバに設定されている環境変数をサーバ・メッセージ・ウィンドウと *server-log.txt* ファイルに出力します。

```
dbeng10 -n myserver -ze -o server-log.txt
```

-zl サーバ・オプション

サーバ上の各データベース接続の、最後に作成された SQL 文の取得をオンにします。

構文

{ dbsrv10 | dbeng10 } -zl …

適用対象

すべてのオペレーティング・システムとサーバ

備考

この機能は、RememberLastStatement サーバ設定を使用してオンにすることもできます。ある接続で最後に準備された SQL 文は、CONNECTION_PROPERTY 関数の LastStatement 値を使用して取得できます。sa_conn_activity ストアド・プロシージャを使用すると、サーバ上の現在のすべてのデータベース接続について、最後に作成された SQL 文を取得できます。

LastStatement の値は、文が準備されると同時に設定され、文が削除されると同時にクリアされます。各接続につき 1 つの文の文字列のみが記憶されます。

ある接続について sa_conn_activity が空でない値を返した場合、その接続で現在実行されている文である可能性が高くなります。その文が完了している場合は、文がすでに削除され、このプロパティの値がクリアされている可能性があります。アプリケーションが複数の文を準備し、それらの文のステートメント・ハンドルを保持している場合、LastStatement が返す値は接続で現在実行されている処理を表しません。

ストアド・プロシージャ・コールの場合、プロシージャ内の文ではなく、最も外側のプロシージャ・コールのみが表示されます。

警告

-zl が指定されている場合、または RememberLastStatement サーバの設定がオンになっている場合は、任意のユーザが sa_conn_activity システム・プロシージャを呼び出すか LastStatement 接続プロパティの値を取得して、他のユーザに対して最後に作成された SQL 文を調べることができます。このオプションは注意して使用し、不要な場合はオフにしてください。

参照

- 「接続レベルのプロパティ」 518 ページ
- 「sa_conn_activity システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
- 「sa_server_option システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』

-zn サーバ・オプション

保管する要求ログ・ファイルのコピー数を指定します。

構文

{ dbsrv10 | dbeng10 } -zn *integer*

適用対象

すべてのオペレーティング・システムとサーバ

備考

要求ロギングが長期間有効になっていると、要求ログ・ファイルのサイズが大きくなることがあります。-zn オプションを使用すると、保管する要求ログ・ファイルのコピー数を指定できます。ただし、-zs も指定されていなければ有効になりません。-zs オプションにより、元のログ・ファイルが指定のサイズに到達すると、新しいログ・ファイルを作成し、元のログ・ファイルの名前を変更できます。「-zs サーバ・オプション」 215 ページを参照してください。

たとえば、要求ロギング情報を *req.out* ファイルにリダイレクトし、-zn オプションを使って 5 つのログ・ファイル・コピーを要求すると、サーバは *req.out.1*、*req.out.2*、*req.out.3*、*req.out.4*、*req.out.5* 順序でファイルを作成します。これらのファイルが存在する場合、アクティブな要求ログが再び一杯になると以下の動作が発生します。

- *req.out.1* が削除される。
- ファル *req.out.2* ー *req.out.5* が *req.out.1* ー *req.out.4* に名前変更される。
- アクティブなログのファイルのコピーが *req.out.5* に名前変更される。

要求ロギングを有効にするには、-zr オプションを使用します。このログは、-zo オプションにより別のファイルにリダイレクトできます。また、sa_server_option システム・プロシージャを使用して、要求ログの数を設定することもできます。その場合、*nn* には、要求ログ・ファイルのコピーの数を指定します。

```
CALL sa_server_option('RequestLogNumFiles',nn)
```

参照

- 「-zo サーバ・オプション」 212 ページ
- 「-zr サーバ・オプション」 213 ページ
- 「-zs サーバ・オプション」 215 ページ
- 「sa_server_option システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』

例

次の例 (全体を 1 行に入力) では、要求ロギング情報は *mydatabase.log* という要求ログ・ファイルへ出力されます。このファイルは最大サイズが 10 KB で、要求ログの 3 つのコピーが保存されます。

```
dbeng10 "c:¥my data¥mydatabase.db" -zr all -zn 3
 -zs 10 -zo mydatabase.log
```

-zo サーバ・オプション

通常のログ・ファイルとは別のファイルに、要求ロギング情報をリダイレクトします。

構文

{ **dbsrv10** | **dbeng10** } **-zo** *filename*…

適用対象

すべてのオペレーティング・システムとサーバ

備考

要求ロギングは、-zr オプションを使用すると有効になります。-zo オプションを指定することによって、出力をこのファイルから通常のログ・ファイルではない別のファイルにリダイレクトできます。

また、このオプションにより、要求ロギングがサーバ・メッセージ・ウィンドウに表示されなくなります。

参照

- 「-zn サーバ・オプション」 211 ページ
- 「-zr サーバ・オプション」 213 ページ
- 「-zs サーバ・オプション」 215 ページ

-zp サーバ・オプション

クエリ・オプティマイザが最近使用したプランの取得をオンにします。

構文

{ dbsrv10 | dbeng10 } -zp …

適用対象

すべてのオペレーティング・システムとサーバ

備考

このオプションは、各接続で最近使用されたクエリ実行プランをデータベース・サーバに保存する場合に使用します。sa_server_option システム・プロシージャの RememberLastPlan サーバ設定を使用して、この機能をオンにすることもできます。LastPlanText 接続プロパティを使用することによって、最近使用されたプランのテキストを表示できます。

参照

- 「接続レベルのプロパティ」 518 ページ
- 「sa_conn_activity システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
- 「sa_server_option システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』

-zr サーバ・オプション

操作の要求ロギングを有効にします。

構文

{ dbsrv10 | dbeng10 } -zr { SQL | HOSTVARS | PLAN | PROCEDURES | TRIGGERS | OTHER | BLOCKS | REPLACE | ALL | YES | NONE | NO } …

適用対象

すべてのオペレーティング・システムとサーバ

備考

このオプションは、問題の原因を突き止める場合にだけ使用します。この情報はサーバ・メッセージ・ウィンドウに表示されるか、ログ・ファイルに送信されます。

-zr の値は、次のような情報を返します。

- **SQL** 以下の項目のロギングを有効にします。
  - START DATABASE 文
  - STOP DATABASE 文
  - STOP ENGINE 文
  - 文の準備と実行
  - EXECUTE IMMEDIATE 文
  - オプション設定
  - COMMIT 文
  - ROLLBACK 文
  - PREPARE TO COMMIT 操作
  - 接続と接続解除
  - トランザクションの開始
  - DROP STATEMENT 文
  - カーソルの説明
  - カーソルを開く、閉じる、再開する
  - エラー
- **PLAN** クエリ・プランのロギングを有効にします (短期)。プロシージャのクエリ・プランは、プロシージャ (PROCEDURES) のロギングが有効な場合にも記録されます。
- **HOSTVARS** ホスト変数の値のロギングを有効にします。HOSTVARS を指定した場合、SQL にリストされている情報もロギングされます。
- **PROCEDURES** プロシージャ内から実行されている文のロギングを有効にします。
- **TRIGGERS** トリガ内から実行されている文のロギングを有効にします。
- **OTHER** SQL に含まれないその他の要求タイプのロギングを有効にします (FETCH や PREFETCH など)。ただし、OTHER を指定して SQL を指定しない場合、SQL+OTHER を指定した場合と同じです。OTHER を含めると、ログ・ファイルが急速に拡大し、サーバのパフォーマンス低下につながることがあります。
- **BLOCKS** 別の接続で接続がブロックされたときと、接続のブロックが解除されたときに表示する詳細のロギングを有効にします。
- **REPLACE** ロギングの開始時に、既存の要求ログは同じ名前を持つ新規の (空の) ログで置換されます。それ以外の場合、既存の要求ログが開き、新規エントリがファイルの末尾に追加されます。

- **ALL**　すべてのサポート情報をロギングします。これは、SQL+PLAN+HOSTVARS +PROCEDURES+TRIGGERS+OTHER+BLOCKS を指定した場合と同じです。この設定では、ログ・ファイルが急速に拡大し、サーバのパフォーマンス低下につながることがあります。
- **NO または NONE**　要求ログに対するロギングを無効にします。

データベース・サーバを起動した後で、sa_server_option システム・プロシージャを使用し、要求ログ設定を変更してロギングの対象とする情報を増減できます。「sa_server_option システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

RequestLogging 設定の現在の値は、次のクエリを使用して検索できます。

```
SELECT PROPERTY( 'RequestLogging' )
```

RequestLogMaxSize
要求ログ情報の記録に使用されるファイルのバイト単位での最大サイズです。-zs 0 を指定した場合は、要求ロギング・ファイルの最大サイズは適用されず、名前は変更されません。これはデフォルト値です。

要求ログ・ファイルが、sa_server_option システム・プロシージャまたは -zs サーバ・オプションで指定されたサイズに達すると、ファイル名が変更されて拡張子 *.old* が追加されます (既存のファイルが存在する場合は、同じ名前で置換されます)。その後、要求ログ・ファイルが再起動されます。「-zs サーバ・オプション」 215 ページを参照してください。

RequestLogNumFiles
保持する要求ログ・ファイルのコピーの数

要求ロギングが長期間有効になっていると、要求ログ・ファイルのサイズが大きくなることがあります。-zn オプションを使用して、保持する要求ログ・ファイルのコピーの数を指定できます。「-zn サーバ・オプション」 211 ページを参照してください。

RequestTiming
データベースに、各要求のタイミング情報を保守するように支持します。この機能はデフォルトでオフになっています。sa_performance_diagnostics プロシージャを使用して、要求のタイミング情報のサマリを取得します。「-zt オプション」 216 ページと「sa_performance_diagnostics システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

SecureFeatures
このデータベース・サーバで実行されるデータベースで無効にする機能を指定します。*feature-list* は、機能名または機能セットのカンマ区切りリストです。有効な *feature-list* 値のリストについては、「-sf サーバ・オプション」 192 ページを参照してください。

参照

- 「-zn サーバ・オプション」 211 ページ
- 「-zo サーバ・オプション」 212 ページ

-zs サーバ・オプション

要求ロギング用のファイルのサイズを制限します。

構文

{ **dbsrv10** | **dbeng10** } **-zs** { *size*[**k** | **m** | **g** } …

適用対象

すべてのオペレーティング・システムとサーバ

備考

要求ロギングを有効にするには、-zr オプションを使用します。このログは、-zo オプションにより別のファイルにリダイレクトできます。また、-zs オプションを使用してファイルのサイズを制限できます。

size には、要求ロギング・ファイルの最大サイズをバイト単位で指定します。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれサフィックス **k**、**m**、または **g** を使用します。

-zs 0 を指定した場合は、要求ロギング・ファイルの最大サイズは適用されず、名前は変更されません。これはデフォルト値です。

要求ログ・ファイルが -zs オプションまたは sa_server_option システム・プロシージャで指定したサイズに到達すると、ファイル名に拡張子 *.old* が追加されて名前が変更されます (同じ名前がある場合、既存のファイルに置き換わります)。要求ログ・ファイルが再起動します。

参照

- 「-zn サーバ・オプション」 211 ページ
- 「-zo サーバ・オプション」 212 ページ
- 「-zr サーバ・オプション」 213 ページ
- 「sa_server_option システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』

例

次の例では、ログ・ファイルのサイズを制御するための -zs オプションの使用方法を示します。次のコマンド・ラインを使用してデータベース・サーバを起動するとします。

```
dbeng10 -zr all -zs 10 -zo mydatabase.log
```

新規ログ・ファイル *mydatabase.log* が作成されます。このファイルのサイズが 10 KB に到達すると、既存の *mydatabase.old* ファイルが削除され、*mydatabase.log* の名前が *mydatabase.old* に変更されて、新しい *mydatabase.log* ファイルが開始されます。このプロセスは、*mydatabase.log* ファイルが指定したサイズ (この場合は 10 KB) に達するたびに繰り返されます。

-zt オプション

要求タイミング情報のロギングをオンにします。

構文

{ **dbsrv10** | **dbeng10** } **-zt** …

適用対象

すべてのオペレーティング・システムとサーバ

備考

データベース・サーバを起動した後で、sa_server_option システム・プロシージャを使用し、要求タイミング情報のロギングのステータスを変更できます。「sa_server_option システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

RequestTiming 設定の現在の値は、次のクエリを使用して検索できます。

```
SELECT PROPERTY( 'RequestTiming' )
```

参照

- 「sa_performance_diagnostics システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
- 「sa_performance_statistics システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』

リカバリ・オプション

これらのオプションは、リカバリの場合にだけ使用されます。

-f リカバリ・オプション

トランザクション・ログが失われた後で、データベース・サーバを強制的に起動します。

構文

{ dbsrv10 | dbeng10 } -f …

適用対象

すべてのオペレーティング・システムとサーバ

備考

トランザクション・ログが存在しない場合、データベース・サーバはデータベースのチェックポイント・リカバリを実行して終了します。データベース・サーバの実行は継続されません。この後、-f オプションを使用せずにデータベース・サーバを再起動し、通常の操作を行うことができます。

データベースと同じディレクトリにトランザクション・ログが存在する場合、データベース・サーバはチェックポイント・リカバリとトランザクション・ログを使ったリカバリを実行して終了します。データベース・サーバの実行は継続されません。この後、-f オプションを使用せずにデータベース・サーバを再起動し、通常の操作を行うことができます。

サーバの起動時にキャッシュ・サイズを指定すると、リカバリ時間を短縮できます。

「バックアップとデータ・リカバリ」 811 ページを参照してください。

例

次のコマンドを入力すると、データベース・サーバがデータベース *mydatabase.db* のリカバリを開始し、実行します。


```
dbeng10 mydatabase.db -f
```

データベース・オプション

次のオプションは、データベース・ファイルの後に指定し、そのデータベースだけに適用されます。

-a データベース・オプション

指定したトランザクション・ログを適用します。-a データベース・オプションは、*database-file* の後に指定する必要があり、そのデータベースだけに適用されます。

構文

{ **dbsrv10** | **dbeng10** } [*server-options*] *database-file* **-a** *log-filename* ⋯

適用対象

すべてのオペレーティング・システムとサーバ

備考

これは、データベース・ファイルのメディア障害からのリカバリに使用されます。このオプションを指定すると、データベース・サーバはログを適用して終了します (実行は継続しません)。複数のトランザクション・ログを適用する場合は、-a オプションを使用するときにトランザクション・ログの正しい適用順序を把握する必要があります。-a の代わりに -ad オプションまたは -ar オプションを使用すると、複数のトランザクション・ログが自動的に正しい順序で適用されます。

サーバの起動時にキャッシュ・サイズを指定すると、リカバリ時間を短縮できます。

「バックアップとデータ・リカバリ」 811 ページを参照してください。

参照

- 「データベース・ファイルのメディア障害からリカバリする」 857 ページ
- 「複数のトランザクション・ログからのリカバリ」 862 ページ
- 「-ad データベース・オプション」 218 ページ
- 「-ar データベース・オプション」 219 ページ

例

次の例 (全体を 1 行に入力) では、ログ・ファイル *demo.log* がサンプル・データベースのバックアップ・コピーに適用されます。

```
dbeng10 "c:¥backup¥demo.db" -a "c:¥backup¥demo.log"
```

-ad データベース・オプション

データベースに適用されるログ・ファイルがあるディレクトリを指定します。-ad データベース・オプションは、*database-file* の後に指定する必要があり、そのデータベースだけに適用されます。

構文

{ **dbsrv10** | **dbeng10** } [*server-options*] *database-file* **-ad** *log-directory* …

適用対象

すべてのオペレーティング・システムとサーバ

備考

-ad オプションを使用すると、指定したディレクトリでデータベースのログ・ファイルが検索されます。ログ・ファイルの開始ログ・オフセットがデータベース・ファイルに保存されている開始ログ・オフセットと等しいか、それより大きい場合、ログ・ファイルはログ・オフセット順に適用されます。すべてのログ・ファイルが適用されると、データベースは停止します。ログ・ファイルの適用が完了した後もデータベースの実行を継続する場合は、-as オプションも指定する必要があります。

参照

- 「データベース・ファイルのメディア障害からリカバリする」 857 ページ
- 「複数のトランザクション・ログからのリカバリ」 862 ページ
- 「-a データベース・オプション」 218 ページ
- 「-ar データベース・オプション」 219 ページ
- 「-as データベース・オプション」 220 ページ

例

次の例では、*backup* ディレクトリ内のログ・ファイルが *mysample.db* データベースに適用され、適用が完了すると、データベースが停止します。

```
dbeng10 "c:¥mysample.db" -ad "c:¥backup"
```

次の例では、*backup* ディレクトリ内のログ・ファイルが *mysample.db* データベースに適用され、適用の完了後もデータベースの実行は継続します。

```
dbeng10 "c:¥mysample.db" -ad "c:¥backup" -as
```

-ar データベース・オプション

トランザクション・ログと同じディレクトリ内にあるログ・ファイルをデータベースに適用します。-ar データベース・オプションは、*database-file* の後に指定する必要があり、そのデータベースだけに適用されます。

構文

{ **dbsrv10** | **dbeng10** } [*server-options*] *database-file* **-ar** …

適用対象

すべてのオペレーティング・システムとサーバ

備考

-ar オプションを使用すると、データベースのログ・ファイルはトランザクション・ログと同じディレクトリで検索されます。トランザクション・ログのディレクトリはデータベースから取得されます。ログ・ファイルの開始ログ・オフセットがデータベースに保存されている開始ログ・オフセットと等しいか、それより大きい場合、ログ・ファイルはログ・オフセット順に適用されます。すべてのログ・ファイルが適用されると、データベースは停止します。ログ・ファイルの適用が完了した後もデータベースの実行を継続する場合は、-as オプションも指定する必要があります。

参照

- 「データベース・ファイルのメディア障害からリカバリする」 857 ページ
- 「複数のトランザクション・ログからのリカバリ」 862 ページ
- 「-a データベース・オプション」 218 ページ
- 「-ad データベース・オプション」 218 ページ
- 「-as データベース・オプション」 220 ページ

例

次の例では、トランザクション・ログ・ファイル (保存されているディレクトリはデータベースから取得されます) が *mysample.db* データベースに適用されます。ログ・ファイルの適用が完了した後もデータベースの実行は継続します。

```
dbeng10 "c:¥mysample.db" -ar -as
```

-as データベース・オプション

トランザクション・ログの適用後もデータベースの実行を継続します (-ad または -ar とともに使用)。-as データベース・オプションは、*database-file* の後に指定する必要があり、そのデータベースだけに適用されます。

構文

{ **dbsrv10** | **dbeng10** } [*server-options*] *database-file* { **-ad** *log-dir* | **-ar** } **-as** …

適用対象

すべてのオペレーティング・システムとサーバ

備考

-as オプションを指定する場合は、-ad オプションまたは -ar オプションも指定する必要があります。-as オプションを指定すると、トランザクション・ログの適用後もデータベースの実行が継続します。

参照

- 「データベース・ファイルのメディア障害からリカバリする」 857 ページ
- 「複数のトランザクション・ログからのリカバリ」 862 ページ
- 「-a データベース・オプション」 218 ページ
- 「-ad データベース・オプション」 218 ページ
- 「-ar データベース・オプション」 219 ページ

例

次の例では、トランザクション・ログが *mysample.db* データベースに適用されます。-ar が指定されているため、データベース・サーバはトランザクション・ログのロケーションをデータベースから取得します。ログ・ファイルの適用が完了した後もデータベースの実行は継続します。

```
dbeng10 "c:¥mysample.db" -ar -as
```

次の例では、*backup* ディレクトリ内のログ・ファイルが *mysample.db* データベースに適用されます。ログ・ファイルの適用が完了した後もデータベースの実行は継続します。

```
dbeng10 "c:¥mysample.db" -ad "c:¥backup" -as
```

-ds データベース・オプション

データベースの DB 領域が配置されているディレクトリを指定します。-ds データベース・オプションは、*database-file* の後に指定する必要があり、そのデータベースだけに適用されます。

構文

{ **dbsrv10** | **dbeng10** } **-ds** *dbspace-directory* ...

適用対象

すべてのオペレーティング・システムとサーバ

備考

DB 領域のディレクトリを指定すると、データベース・サーバは、DB 領域を探すときにこのディレクトリだけを検索対象とします。DB 領域のロケーションは、サーバ・メッセージ・ウィンドウに表示されます。

フル・パス名を指定された DB 領域がバックアップに含まれている場合は、このオプションを使用して、元のデータベースが実行中である間に、元のデータベースと同じコンピュータ上にあるデータベースのバックアップ・コピーを起動できます。

参照

- 「追加 DB 領域の使用」 294 ページ
- 「START DATABASE 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「STOP DATABASE 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「default_dbspace オプション [データベース]」 444 ページ

例

次の例では、ディレクトリ *c:¥backup¥Nov15* に対して DB 領域を検索するデータベース・サーバを起動します。

```
dbeng10 c:¥backup¥Nov15¥my.db -ds c:¥backup¥Nov15¥
```

次の例では、現在のディレクトリに対して DB 領域を検索するデータベース・サーバを起動します。

```
dbeng10 my.db -ds .
```


-dh データベース・オプション

このオプションを指定すると、サーバに対してサーバ列挙ユーティリティ (dblocate) を実行した場合、データベースが表示されません。-dh データベース・オプションは、*database-file* の後に指定する必要があり、そのデータベースだけに適用されます。

構文

{ **dbsrv10** | **dbeng10** } [*server-options*] *database-file* **-dh** …

適用対象

すべてのプラットフォーム

備考

-dh オプションを指定すると、サーバに対してサーバ列挙ユーティリティ (dblocate) を実行したときに、データベースが検出されません。そのため、-d、-dn、-dv のいずれかのオプションを指定して dblocate を実行した場合、-dh オプションを指定したデータベースは表示されません。

参照

- 「サーバ列挙ユーティリティ (dblocate)」 698 ページ

-ek データベース・オプション

強力に暗号化されたデータベースのキーを指定します。-ek データベース・オプションは、*database-file* の後に指定する必要があり、そのデータベースだけに適用されます。

構文

{ **dbsrv10** | **dbeng10** } [*server-options*] *database-file* **-ek** *key* …

備考

暗号化されたデータベースを起動するには、key 値を -ek オプションに指定してください。key は、大文字、小文字、数字、文字、特殊記号を含む文字列です。

クリア・テキストで見ることができないように暗号化キーをダイアログに入力するには、-ep サーバ・オプションを使用します。「-ep サーバ・オプション」 164 ページを参照してください。

クライアント・アプリケーションとデータベース・サーバ間の通信パケットを安全化するには、-ec サーバ・オプションとトランスポート・レイヤ・セキュリティを使用します。「トランスポート・レイヤ・セキュリティ」 949 ページを参照してください。

参照

- 「-ep サーバ・オプション」 164 ページ
- 「DatabaseKey 接続パラメータ [DBKEY]」 242 ページ

例

次の例では、データベースを起動して、コマンド・ラインで暗号化キーを指定します。

```
dbsrv10 -x tcpip mydata.db -ek "Akmm9u70y"
```

-m データベース・オプション

チェックポイントの実行後にトランザクション・ログの内容を削除します。-m データベース・オプションは、*database-file* の後に指定する必要があり、そのデータベースだけに適用されます。

構文

{ **dbsrv10** | **dbeng10** } [*server-options*] *database-file* **-m** …

適用対象

すべてのオペレーティング・システムとサーバ

備考

シャットダウン時、またはサーバでスケジュールされたチェックポイントの結果としてチェックポイントが実行されたときに、トランザクション・ログを削除します。これでトランザクション・ログの肥大化が自動的に制限されます。チェックポイントの頻度は、checkpoint_time と recovery_time オプションによって制御できます (また、データベース・サーバの -gc と -gr コマンド・ライン・オプションでも設定できます)。

-m オプションは、高速な応答時間を必要とする大容量のトランザクションを処理する場合や、リカバリやレプリケーションがトランザクション・ログの内容に依存しない場合に役立ちます。このオプションを選択すると、データベース・ファイルを含むデバイスのメディア障害に対して無防備な状態になります。

データベース・ファイルの断片化を防ぐためには、このオプションを使用する場合に、トランザクション・ログをデータベースそのものとは別のデバイスまたはパーティションに保管することをおすすめします。

このオプションは -m サーバ・オプションと同じですが、現在のデータベースまたは *database-file* 変数で識別されるデータベースにのみ適用されます。

レプリケートされるデータベース
レプリケートされるデータベースでは -m オプションを使わないでください。レプリケーションは本質的にトランザクション・ログ情報に依存します。

例

次の例では、silver という名前のデータベース・サーバが起動され、データベース *salesdata.db* がロードされます。チェックポイントが終了すると、トランザクション・ログの内容が削除されます。

```
dbsrv10 -n silver "c:¥inventory details¥salesdata.db" -m
```

-n データベース・オプション

データベースの名前を設定します。-n データベース・オプションは、*database-file* の後に指定する必要があり、そのデータベースだけに適用されます。

構文

{ **dbsrv10** | **dbeng10** } [*server-options*] *database-file* **-n** *string* …

適用対象

すべてのオペレーティング・システムとサーバ

備考

データベース・サーバとデータベースはどちらも名前を付けることができます。データベース・サーバはいくつかのデータベースをロードできるので、データベース名を使用して、各データベースを区別します。

デフォルトでは、データベースはパスと拡張子を除いたデータベースのファイル名を受け取ります。たとえば、-n オプションを指定しないでデータベース *samples-dir¥demo.db* を起動すると、データベース名は demo になります。

データベースには、以下に該当する名前を付けることはできません。

- 空白スペース、一重引用符、または二重引用符で始まる名前
- 空白スペースで終わる名前
- セミコロンを含む名前

例

次の例では、データベース・サーバがキャッシュ・サイズ 3 MB で起動され、データベースがロードされます。データベースには test という名前が付けられます。データベース・サーバ名が指定されていないため、サーバは最初のデータベースから名前を取得します。そのため、サーバの名前も test になります。

```
dbsrv10 -c 3MB "c:¥mydata.db" -n "test"
```

2 つの -n オプション
-n オプションの内容は指定する位置によって異なります。データベース・ファイル名の前に指定すると、サーバ・オプションとしてサーバを指定します。データベース・ファイル名の後に指定すると、データベース・オプションとしてデータベースを指定します。
たとえば、次のコマンドでは、サーバ SERV とデータベース DATA が指定されます。

```
dbsrv10 -n SERV c:¥mydata.db -n DATA
```

「-n サーバ・オプション」 181 ページを参照してください。

-r データベース・オプション

指定されたデータベースを読み込み専用として起動します。データベースへの変更はできません。つまり、サーバはデータベース・ファイルやトランザクション・ログ・ファイルを変更しません。-r データベース・オプションは、*database-file* の後に指定する必要があり、そのデータベースだけに適用されます。

構文

{ **dbsrv10** | **dbeng10** } [*server-options*] *database-file* **-r** …

適用対象

すべてのオペレーティング・システムとサーバ

備考

コマンド・ラインでどのデータベース名よりも前にオプションを指定した場合、テンポラリ・ファイルを除くすべてのデータベース・ファイル (メイン・データベース・ファイル、DB 領域、トランザクション・ログ、トランザクション・ログ・ミラー) が読み込み専用モードで開きます。あるデータベース名の後ろに -r サーバ・オプションを指定した場合、そのデータベースだけが読み込み専用になります。テンポラリ・テーブルに変更を加えることはできますが、トランザクション・ログとロールバック・ログが無効化されているため、ROLLBACK を実行しても効果はありません。

変更が不可能なデータベース・ファイルの例として、CD-ROM によって配賦されるデータベースがあります。このようなデータベースには読み込み専用モードでアクセスできます。

INSERT 文や DELETE 文などを使ってデータベースを変更しようとすると、SQLSTATE_READ_ONLY_DATABASE エラーが発生します。

リカバリを必要とするデータベースは、読み込み専用モードでは起動できません。たとえば、オンライン・バックアップを使って作成したデータベース・ファイルは、バックアップの開始時に開いているトランザクションがあると、読み込み専用モードで起動できません。これは、バックアップ・コピーの開始時に、これらのトランザクションがリカバリを必要とするためです。

監査がオンである場合、データベースを読み込み専用モードで起動することはできません。

参照

- 「-r サーバ・オプション」 189 ページ
- 「auditing オプション [データベース]」 428 ページ

例

2 つのデータベースを読み込み専用モードで開くには、次のように指定します。

```
dbeng10 -r database1.db database2.db
```

2 つのうち最初のデータベースだけを読み込み専用モードで開くには、次のように指定します。

```
dbeng10 database1.db -r database2.db
```

-sn オプション

データベース・サーバ上で動作する 1 つのデータベースに代替サーバ名を割り当てます。-sn データベース・オプションは、*database-file* の後に指定する必要があり、そのデータベースだけに適用されます。

構文

{ **dbsrv10** | **dbeng10** } [*server-options*] *database-file* **-sn** *alternate-server-name*

適用対象

すべてのオペレーティング・システム、ネットワーク・サーバのみ

備考

データベース・サーバが特定のデータベース・サーバに対応する複数のサーバ名を受信するように設定できます。実サーバ名以外のサーバ名を代替サーバ名といいます。代替サーバ名はデータベース・サーバ上の特定のデータベースに固有のものです。クライアントが代替サーバ名を使用して接続できるのは、その代替サーバ名が指定されているデータベース*だけ*です。

代替サーバ名はネットワーク全体でユニークでなければなりません。代替サーバ名がユニークでない場合、データベースの起動は失敗します。データベースをサーバ・コマンドで起動し、代替サーバ名がユニークでない場合、サーバの起動は失敗します。START DATABASE 文を使用して代替サーバ名を指定することもできます。

代替サーバ名を指定するクライアントが接続できるのは、その代替サーバ名が指定されたデータベースだけです。同じデータベース・サーバ上の他のデータベースには接続できません。接続パラメータの DBN または DBF を指定する場合は、それぞれデータベース名またはデータベース・ファイルと一致している必要があります。接続パラメータの DBN または DBF を指定しない場合、データベースはそのサーバのデフォルト・データベースと同じように動作します。

サーバ列挙ユーティリティ (dblocate) では、代替サーバ名も検出されます。

データベース・ミラーリングでの代替サーバ名の使用
データベース・ミラーリングを使用している場合は、クライアント・アプリケーションがプライマリ・サーバとミラー・サーバを事前に把握していなくても現在のプライマリ・サーバに接続できるように、代替サーバ名を*指定する必要があります*。両方の稼働しているサーバでは、代替サーバ名として同じ名前を使用することが*必要*です。

参照

- 「-xa サーバ・オプション」 207 ページ
- 「-xf サーバ・オプション」 207 ページ
- 「-xp データベース・オプション」 227 ページ
- 「START DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「データベース・ミラーリングの概要」 888 ページ
- 「サーバ列挙ユーティリティ (dblocate)」 698 ページ

例

次のコマンドは、データベース・サーバ myserver 上のデータベース *satest.db* と *sample.db* を起動します。-sn オプションによって、*sample.db* に接続するときにデータベース・サーバが代替サーバ名として mysample が使用されます。

```
dbsrv10 -n myserver satest.db sample.db -sn mysample
```

sample.db に接続するときには、次の接続パラメータを使用できます。

- ENG=myserver;DBN=sample
- ENG=mysample
- ENG=mysample;DBN=sample

ENG=mysample を使用して *satest.db* に接続することはできません。

-xp データベース・オプション

データベース・ミラーリングが使用されている場合に、稼働しているサーバにパートナーと監視サーバへの接続を可能にする情報を提供します。-xp データベース・オプションは、*database-file* の後に指定する必要があり、そのデータベースだけに適用されます。

構文

{ **dbsrv10** | **dbeng10** } [*server-options*] *database-file*
-xp partner=(*partner-conn*);
auth=*auth-str*;
[;**arbiter=**(*arbiter-conn*)]
[;**mode=**[**sync** | **async** | **page**]
[;**autofailover=**[**YES** | **NO**]]
[;**pagetimeout=***n*]
[; **preferred=**[**YES** | **NO**] ⋯

適用対象

すべてのオペレーティング・システム (Windows CE を除く)、ネットワーク・サーバのみ

備考

-xp を指定する場合、-xf オプションも指定して、データベース・ミラーリングのステータス情報ファイルのロケーションを指定する必要があります。

partner-conn　パートナー・サーバの接続文字列を指定します。ユーザ ID とパスワードは必要ありません。フェールオーバ時間を短縮するためにタイムアウトを指定するようおすすめします。

auth-str　監視サーバが使用する認証文字列を指定します。

arbiter-conn　監視サーバの接続文字列を指定します。ユーザ ID とパスワードは必要ありません。フェールオーバ時間を短縮するためにタイムアウトを指定するようおすすめします。

mode　データベース・ミラーリングで使用する同期実行モードとして、同期 (sync)、非同期 (async)、非同期フルページ (page) のいずれかを指定します。

autofailover　プライマリ・サーバで障害が発生した場合にミラー・サーバが自動的にプライマリ・サーバになるかどうかを指定します。このオプションは同期モードには適用されません。

> **注意**
> 非同期モードまたは非同期フルページ・モードを使用している場合は、-xp autofailover オプションを Yes に設定することをおすすめします。それによって、プライマリ・サーバで障害が発生した場合、ミラー・サーバが自動的にプライマリ・サーバとなります。

pagetimeout　トランザクション・ログ・ページを満杯かどうかにかかわらずミラー・サーバに送信する間隔を秒単位で指定します。このオプションは、非同期フルページ・モードを使用している場合のみ適用されます。

preferred　サーバがミラーリング・システムにおいて優先サーバであるかどうかを指定します。優先サーバは、可能なかぎり、プライマリ・サーバのロールを引き受けます。「優先データベース・サーバの指定」 901 ページを参照してください。

参照

- 「データベース・ミラーリングで使用するモードの選択」 892 ページ
- 「-sn オプション」 225 ページ
- 「-xa サーバ・オプション」 207 ページ
- 「-xf サーバ・オプション」 207 ページ

例

次のコマンドは、パートナー・サーバ server2 と監視サーバ arbsrv のパラメータを指定します。

```
dbsrv10 -n server1 mydata.db -sn mydata
-xp partner=(ENG=server2;LINKS=tcpip(TIMEOUT=1));
AUTH=abc;arbiter=(ENG=arbsrv;LINKS=tcpip(TIMEOUT=1))
```

第 6 章

接続パラメータとネットワーク・プロトコル・オプション

目次

接続パラメータ

この項では、各接続パラメータについて説明します。接続パラメータは、接続文字列に含めます。次の場所で入力できます。

- アプリケーションの接続文字列。「接続パラメータ・リストのアセンブル」 90 ページと「接続文字列として渡される接続パラメータ」 62 ページを参照してください。
- ODBC データ・ソース。「ODBC データ・ソースの使用」 75 ページを参照してください。
- SQL Anywhere の [接続] ダイアログ。「SQL Anywhere ユーティリティからの接続」 73 ページを参照してください。

[ODBC 設定] ダイアログと Windows オペレーティング・システム用の SQL Anywhere の [接続] ダイアログは、フォーマットが共通です。一部のパラメータは、これらのダイアログのチェックボックスやフィールドに対応しています。その他のパラメータは、[詳細] タブにあるテキスト・ボックスに入力できます。

注意

- 値に大文字と小文字が区別されるもの (UNIX のファイル名など) が含まれている場合でも、接続パラメータは大文字と小文字を区別しません。
- ブール・パラメータは、YES、Y、ON、TRUE、T、1 のいずれかによってオンになり、NO、N、OFF、FALSE、F、0 のいずれかによってオフになります。パラメータは、大文字と小文字を区別しません。
- 各接続パラメータの使用法のところで、パラメータが使用される状況を説明します。一般的に使用法にあげる項目は以下のとおりです。
  - **組み込みデータベース**　SQL Anywhere を組み込みデータベースとして使用した場合、接続するとパーソナル・サーバが起動し、データベースがロードされます。アプリケーションがデータベースから切断されると、データベースはアンロードされ、サーバが停止します。
  - **実行中のローカル・データベース**　これは、SQL Anywhere パーソナル・サーバがすでに実行中で、データベースがすでにサーバにロードされている場合を指します。
  - **ネットワーク・サーバ**　SQL Anywhere をネットワーク・サーバとして使用する場合、クライアント・アプリケーションはネットワーク上ですでに実行しているサーバを検出し、データベースに接続します。
- dbping ユーティリティを使用して、接続文字列をテストできます。たとえば、demo10 という名前のパーソナル・サーバがサンプル・データベース (コマンド dbeng10 samples-dir ¥demo.db で実行可能) を実行しているとします。次の文字列は、ローカル・コンピュータ上で demo10 という名前のデータベース・サーバが実行中であり、demo という名前のデータベースが実行されているときに、「**データベースへの ping が成功しました。**」というメッセージを返します。

```
dbping -d -c "ENG=demo10;DBN=demo;UID=DBA;PWD=sql"
```

ただし、次のコマンドは、ローカル・コンピュータ上で other-server という名前のデータベース・サーバが実行中ではない場合、「**データベースへの ping が失敗しました -- データベース・サーバは起動していません。**」というメッセージを返します。

```
dbping -d -c "ENG=other-server;UID=DBA;PWD=sql"
```

「Ping ユーティリティ (dbping)」 693 ページを参照してください。

参照

- 「接続パラメータについての注意」 86 ページ

AppInfo 接続パラメータ [APP]

データベース・サーバからの特定のクライアント接続の開始を、管理者が容易に識別できるようにします。

使用法

特に制限なし

値の範囲

文字列

デフォルト

空の文字列

備考

この接続パラメータは、Embedded SQL、ODBC、OLE DB、または ADO.NET クライアント、および iAnywhere JDBC ドライバを使用するアプリケーションから、データベース・サーバに送信されます。Open Client または jConnect アプリケーションから使用することはできません。

このパラメータは、クライアント・コンピュータの IP アドレスや実行されているオペレーティング・システムなどの、クライアント・プロセスについての情報を保持するように生成された文字列で構成されています。文字列は、接続するデータベース・サーバに関連付けられており、次の文を使用して検索できます。

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

クライアントは、固有の文字列も指定できます。指定した文字列は、生成された文字列に追加されます。AppInfo プロパティ文字列は、セミコロンで区切られた **key**=*value* ペアのシーケンスです。有効なキーは次のとおりです。

- **API**　DBLIB、ODBC、OLEDB、ADO.NET、iAnywhereJDBC、PHP、PerlDBD、DBEXPRESS のいずれか
- **APPINFO**　接続文字列に AppInfo を指定したときに入力される文字列
- **EXE**　クライアント実行プログラムの名前 (Windows と NetWare のみ)
- **HOST**　クライアント・コンピュータのホスト名

- **IP**　クライアント・コンピュータの IP アドレス
- **OS**　オペレーティング・システム名とバージョン番号 (例、Windows 2000、NetWare 5.1)
- **PID**　クライアントのプロセス ID (Windows と UNIX のみ)
- **THREAD**　クライアントのスレッド ID (Windows と UNIX のみ)
- **TIMEZONEADJUSTMENT**　接続のローカル時間を表示するために協定世界時 (UTC: Coordinated Universal Time) に加算する必要がある分数
- **VERSION**　主要なバージョン番号、それ以外のバージョン番号、ビルド番号を含む、使用しているクライアント・ライブラリのバージョン (例、10.0.0.2023)

クライアント接続パラメータにデバッグ・ログ・ファイルを指定すると、そのファイルに APPINFO 文字列が追加されます。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ
- 「request_timeout オプション [データベース]」 490 ページ

例

Interactive SQL からのサンプル・データベースに接続します (デフォルトで iAnywhere JDBC ドライバを使用します)。

```
dbisql -c "UID=DBA;PWD=sql;DBF=samples-dir¥demo.db"
```

アプリケーション情報を表示します。

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

結果は、次のとおりです (1 つの文字列で表示されます)。

```
IP=ip-address;HOST=computer-name;OS='Windows 2000 Build 2195 Service Pack
3';PID=0x724;THREAD=0x6bc;EXE=c:¥Program Files¥SQL Anywhere 10¥win32
¥dbisqlg.exe;VERSION=10.0.0.2023;API=iAnywhereJDBC;TIMEZONEADJUSTMENT=-300
```

AppInfo プロパティにユーザ固有の情報を追加して、Interactive SQL からサンプル・データベースに接続します。

```
dbisql -c "UID=DBA;PWD=sql;DBF=samples-dir¥demo.db;APP=Interactive SQL connection"
```

アプリケーション情報を表示します。

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

結果は、次のとおりです (1 つの文字列で表示されます)。

```
IP=ip-address;HOST=computer-name;OS='Windows 2000 Build 2195 Service Pack
3';PID=0x8d0;THREAD=0xd74;EXE=c:¥Program Files¥SQL Anywhere 10¥win32
¥dbisqlg.exe;VERSION=10.0.0.2023;API=iAnywhereJDBC;TIMEZONEADJUSTMENT=-300;APPINFO=
ISQL connection
```

AutoStart 接続パラメータ [ASTART]

接続が検出できない場合にローカル・データベース・サーバを起動するかどうかを制御します。

使用法

特に制限なし

値の範囲

YES、NO

デフォルト

YES

備考

データベース・ファイル、データベース名、START 接続パラメータのいずれかが指定されていて、接続時にサーバが検出できない場合、デフォルトでは、データベース・サーバは同一のコンピュータで起動されます。接続文字列の AutoStart (ASTART) 接続パラメータを NO に設定することによって、このような動作を無効にできます。CommLinks [LINKS] パラメータに TCPIP または SPX が含まれる場合、データベース・サーバは自動的には起動されません。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ
- 「CommLinks 接続パラメータ [LINKS]」 235 ページ

AutoStop 接続パラメータ [ASTOP]

オープン接続がなくなった場合に、すぐにデータベースを停止するかどうかを制御します。

使用法

組み込みデータベース

値の範囲

YES、NO

デフォルト

YES

備考

デフォルトでは、接続文字列で起動したすべてのサーバは、接続がなくなると停止します。また、接続文字列からロードしたすべてのデータベースも、接続がなくなるとすぐにアンロードされます。この動作は、AutoStop=YES と同じです。

AutoStop=NO を指定すると、その接続で起動したすべてのデータベースは、接続がなくなっても実行を続けます。したがって、データベース・サーバは操作可能な状態を維持します。

AutoStop (ASTOP) 接続パラメータは、現在実行していないデータベースに接続するときにのみ使用されます。データベースがすでに起動されている場合は、無視されます。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

CharSet 接続パラメータ [CS]

この接続で使用する文字セットを指定します。

使用法

特に制限なし

値

文字列

デフォルト

ローカル文字セット

ローカル文字セットがどのように決定されるかについては、「ロケール情報の確認」 354 ページを参照してください。

備考

CharSet に値を指定すると、指定された文字セットが現在の接続に使用されます。設定 CharSet=none は、接続の文字セット変換を無効にします。

データをアンロードする場合は、CharSet 接続パラメータを使用して文字セットを指定します。有効な文字セット値の詳細については、「推奨文字セットと照合」 361 ページを参照してください。

ユニコード・クライアント API を使用している場合は、文字セット変換によるデータの損失を避けるために、CHARSET 接続パラメータを設定することはおすすめできません。ユニコード・クライアント API には、ADO.NET、OLE DB、iAnywhere JDBC ドライバが含まれます。ODBC も、ワイド (ユニコード) 関数が使用されている場合は、ユニコード・クライアント API です。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

CommBufferSize 接続パラメータ [CBSIZE]

通信パケットの最大サイズをバイト単位で設定します。

使用法

特に制限なし

値

整数 [**k**]

デフォルト

CommBufferSize 値が設定されていない場合、CommBufferSize は、サーバ側の設定によって制御されます。デフォルトは 1460 バイトです。

備考

CommBufferSize (CBSIZE) 接続パラメータは、通信パケットのサイズをバイトで指定します。キロバイトの単位を指定するには、**k** を使用します。CommBufferSize の最小値は 300 バイトで、最大値は 16000 バイトです。

ネットワーク上のパケットの最大サイズは、プロトコル・スタックによって設定されます。CommBufferSize をネットワークで許可されているサイズより大きく設定すると、最も大きいバッファがネットワーク・ソフトウェアによって分割されます。ネットワーク・ソフトウェアは、ネットワーク経由で送信する前に各バッファに情報を追加することがあるため、バッファ・サイズをネットワークで許可されているサイズよりいくらか小さく設定してください。TCP/IP を使用している場合、デフォルト値の 1460 で Ethernet パケットに十分対応できます。

パケット・サイズを大きくすると、複数のローのフェッチと長いローのフェッチのパフォーマンスが向上しますが、クライアントとサーバのメモリ使用量が増加します。

クライアント側で CommBufferSize の指定がないと、接続ではサーバのバッファ・サイズが使用されます。クライアント側で CommBufferSize の指定がある場合、接続では CommBufferSize 値が使用されます。

-p データベース・サーバ・オプションを使用して CommBufferSize を設定すると、CommBufferSize を指定していないすべてのクライアントで -p データベース・サーバ・オプションで指定されたサイズが使用されます。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

バッファ・サイズを 400 バイトに設定します。

```
...
CommBufferSize=400
...
```

別の方法として、[ODBC の設定] ダイアログの [ネットワーク] タブにある [バッファ・サイズ] テキスト・ボックスに値を入力して、このパラメータを設定することもできます。

CommLinks 接続パラメータ [LINKS]

クライアント側のネットワーク・プロトコル・オプションを指定します。

使用法

特に制限なし。CommLinks (LINKS) 接続パラメータは、パーソナル・サーバへの接続ではオプションですが、ネットワーク・サーバへの接続では必須です。

値

文字列

デフォルト

接続には共有メモリ通信プロトコルのみを使用します。

備考

CommLinks (LINKS) 接続パラメータを指定しないと、クライアントは現在のコンピュータにあるサーバしか検索せず、共有メモリ接続のみを使用します。これはデフォルトの動作であり、CommLinks=ShMem を指定するのと同じです。共有メモリ・プロトコルは、パーソナル・データベース・サーバに接続するアプリケーションでの標準的な使い方で、同じコンピュータで実行されているクライアントとサーバ間で最速の通信リンクです。

UNIX の共有メモリ接続の保護の詳細については、「セキュリティのヒント」 920 ページを参照してください。

CommLinks=ALL と指定すると、クライアントは、使用可能なすべての通信プロトコルを使用してサーバを検索します。CommLinks=ALL を指定するとパフォーマンスに影響する場合があるため、この設定は使用するプロトコルが不明なときにのみ使用してください。

CommLinks (LINKS) 接続パラメータに 1 つ以上のプロトコルを指定すると、クライアントは、指定された通信プロトコルを使用して、*指定された順番*でネットワーク・データベース・サーバを検索します。共有メモリが指定された場合は、まず共有メモリを使用する接続が試行され、その後にその他の接続プロトコルが指定されている順序で試行されます。指定したプロトコルを使用した接続が失敗すると、試行リストにプロトコルが残っていても、接続エラーが表示されて接続の試行がアボートされます。

CommLinks (LINKS) 接続パラメータの値は、大文字と小文字を区別しません。次の値が含まれます。

- **SharedMemory (ShMem)** 同一コンピュータ通信の共有メモリ・プロトコルを起動します。これはデフォルト設定です。共有メモリがプロトコルのリストに指定されている場合は、リストでの順序に関係なく、クライアントは最初に共有メモリを使用しようとします。
- **ALL** 最初に共有メモリ・プロトコルを使用して接続を試行し、次に使用可能なすべての通信プロトコルを使用します。使用する通信プロトコルが不明の場合は、この設定を使用してください。
- **TCPIP (TCP)** TCP/IP 通信プロトコルを起動します。TCP/IP は、すべてのオペレーティング・システムでサポートされています。
- **SPX** SPX 通信プロトコルを起動します。SPX プロトコルは Windows と NetWare クライアントでサポートされています。

これらの値には、それぞれ追加のネットワーク・プロトコル・オプションを指定できます。

「ネットワーク・プロトコル・オプション」 265 ページを参照してください。

次のような理由がある場合は、ALL ではなく、特定のプロトコルを使用できます。

- クライアントが必要なネットワーク・プロトコルのみを使用すると、ネットワーク・ライブラリの起動時間が少し短縮される。
- データベースへの接続が、速い場合がある。
- 追加のネットワーク・プロトコル・オプションを指定して特定のプロトコルのブロードキャスト動作をチューニングする場合は、明示的にプロトコルを指定する必要がある。

CommLinks (LINKS) 接続パラメータは、データベース・サーバの -x オプションに対応します。

参照

- 「ネットワーク・プロトコル・オプション」 265 ページ
- 「クライアント／サーバ通信」 121 ページ
- 「-x サーバ・オプション」 205 ページ
- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

次の接続文字列フラグメントでは、TCP/IP プロトコルのみを起動します。

```
CommLinks=tcpip
```

次の接続文字列フラグメントは、共有メモリ・プロトコルを起動し、共有メモリ上でデータベース・サーバを検索します。検索が失敗すると、TCP/IP リンクを起動し、ローカル・ネットワーク上のサーバを検索します。サーバの検索が失敗すると、SPX リンクを起動し、SPX 経由でサーバを検索します。

```
CommLinks=tcpip,shmem,spx
```

次の接続文字列フラグメントでは、SPX ポートを起動し、SPX 経由でサーバを検索します。検索が失敗すると、TCP リンクを起動し、ローカル・ネットワーク上のサーバと、ホスト kangaroo を検索します。SPX 上でサーバが検出された場合、TCP リンクは起動 *されません*。

```
CommLinks=spx,tcpip(HOST=kangaroo)
```

Compress 接続パラメータ [COMP]

接続の圧縮をオンまたはオフに設定します。状況によっては、接続を圧縮することでパフォーマンスが向上する場合があります。

使用法

TDS 接続以外。他には特に制限なし。TDS 接続 (jConnect を含む) では、SQL Anywhere 通信圧縮はサポートされません。

値

YES、**NO**

クライアントとサーバで設定が一致しない場合、クライアント側の設定が適用されます。

デフォルト

NO

Compress 接続パラメータに値が設定されていない場合、圧縮ステータスはサーバ側の設定によって制御されます。デフォルトでは、圧縮を行いません。

備考

SQL Anywhere クライアントとサーバの間でやり取りされるパケットは、Compress (COMP) 接続パラメータを使用して圧縮できます。大幅に圧縮可能なデータの大規模なデータ転送では、圧縮率が高くなります。

YES または NO を指定して、この接続の通信圧縮をオンまたはオフに設定します。大文字と小文字は区別されません。

特定のアプリケーションを使用してネットワークのパフォーマンス分析を行ってから、運用環境で通信の圧縮を使用することをおすすめします。

サーバのすべてのリモート接続で圧縮を有効にするには、-pc サーバ・オプションを使用します。

-pc オプションまたは COMPRESS=YES パラメータを指定しても、使用する通信リンクに関わらず、同一コンピュータ接続では圧縮は有効にならないことに注意してください。

参照

- 「-pc サーバ・オプション」 185 ページ
- 「パフォーマンス改善のための通信圧縮設定の調整」 130 ページ
- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

次の接続文字列フラグメントは、パケット圧縮をオンに設定します。

```
Compress=YES
```

次の接続文字列フラグメントは、パケット圧縮をオフに設定します。

```
Compress=NO
```

CompressionThreshold 接続パラメータ [COMPTH]

パケットの圧縮が適用される最小パケット・サイズを増減します。圧縮した場合に転送速度が速くなるパケットのみを圧縮するように圧縮スレッショルドを変更して、圧縮接続のパフォーマンスを向上できます。

使用法

TDS 以外。他には特に制限なし。圧縮接続にのみ適用。

値

整数 [**k**]

クライアントとサーバで圧縮スレッショルドの設定が異なる場合は、クライアントの設定が適用されます。

デフォルト

120

CompressionThreshold 値が設定されていない場合、圧縮スレッショルド値は、サーバ側の設定によって制御されます。デフォルトは 120 バイトです。

備考

圧縮が有効な場合、パケットは、各々のサイズに応じて圧縮するかどうかを決定します。たとえば、SQL Anywhere では、圧縮のスレッショルドよりも小さいパケットは、通信の圧縮が有効な場合でも圧縮されません。同様に、小さなパケット (100 バイト未満) は、通常はまったく圧縮されません。パケットの圧縮には CPU 時間が必要なので、小さなパケットを圧縮しようとすると、実際にパフォーマンスが低下することがあります。

圧縮するパケットの最小サイズをバイト単位で示す値。キロバイトの単位を指定するには、**k** を使用します。サポートされている最小値は 1 バイト、最大値は 32767 バイトです。80 バイト未満の値はおすすめしません。

一般的に、圧縮スレッショルド値を低く設定すると、伝送速度が非常に遅いネットワークのパフォーマンスが向上し、圧縮スレッショルド値を高く設定すると、CPU の消費量が減ってパフォーマンスが向上することがあります。ただし、圧縮のスレッショルド値を小さくするとクライアントとサーバの両方で CPU 使用率が増加するので、パフォーマンス分析を行って、圧縮のスレッショルドを変更するとパフォーマンスが向上するかどうかを判断してください。

参照

- 「-pt サーバ・オプション」 186 ページ
- 「パフォーマンス改善のための通信圧縮設定の調整」 130 ページ
- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

圧縮スレッショルド値を 100 バイトに設定して接続します。

```
CompressionThreshold=100
```

ConnectionName 接続パラメータ [CON]

接続に名前を付け、マルチ接続アプリケーションで簡単に切り替えができるようにします。

使用法

特に制限なし

値

文字列

デフォルト

接続名なし

備考

確立中の特定の接続に名前を付けるオプションのパラメータです。複数の接続を確立しても切り替えを行わないときは、このパラメータを指定する必要はありません。

接続名はデータ・ソース名とは異なります。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ
- 「SET CONNECTION statement [Interactive SQL] [ESQL]」 『SQL Anywhere サーバ - SQL リファレンス』

例

first-con という名前の接続を指定して接続します。

```
CON=first-con
```

DatabaseFile 接続パラメータ [DBF]

まだ実行していないデータベースを起動したときにロードして接続するデータベース・ファイルを指定します。

すでに実行中のデータベースに接続する場合は、DatabaseName (DBN) パラメータを使用します。

使用法

組み込みデータベース

値

文字列

デフォルト

デフォルト設定なし

備考

DatabaseFile (DBF) 接続パラメータは、まだデータベース・サーバで実行していない特定のデータベース・ファイルのロードと接続を行うために使用します。

- 接続したいデータベースがまだ実行されていない場合に DatabaseFile (DBF) 接続パラメータを使用すると、データベースを起動できます。
- ファイル名に拡張子がない場合は、SQL Anywhere が *.db* 拡張子のついたファイルを検索します。
- ファイルのパスは、データベース・サーバの作業ディレクトリの相対パスです。サーバをコマンド・プロンプトから起動すると、コマンド入力時の (現在の) ディレクトリが作業ディレ

クトリになります。サーバをアイコンかショートカットから起動すると、アイコンかショートカットを指定したディレクトリが作業ディレクトリになります。完全なパスとファイル名を指定することをおすすめします。

- データベース・ファイルとデータベース名の両方を指定すると、指定した名前の実行中のデータベースに接続します (データベース・ファイルは無視されます)。接続に失敗すると、データベース・ファイルとデータベース名の両方を使用して、データベースが自動的に起動されます。CommLinks [LINKS] パラメータに TCPIP または SPX が含まれる場合、データベース・サーバは自動的には起動されません。

UNC ファイル名や Novell NetWare Directory Services のファイル名も使用できます。

UNC ファイル名と Novell NetWare Directory Services ファイル名の使用の詳細については、「SQL Anywhere データベース・サーバ」 138 ページを参照してください。

まだ実行していないデータベース・ファイルを自動的に起動する場合、配備されたアプリケーションでは、EngineName (ENG) パラメータを使用してデータベース・サーバ名を指定することをおすすめします。そうしないと、アプリケーションが別のデータベース・サーバに接続する可能性があります。たとえば、データベース・サーバは、すでに実行されている埋め込みアプリケーションの一部である、異なるバージョンの SQL Anywhere サーバに接続することがあります。

警告
データベース・ファイルは、データベース・サーバと同じコンピュータ上に置いてください。ネットワーク・ドライブにあるデータベース・ファイルを起動すると、ファイルが破損することがあります。

参照

- 「-gd サーバ・オプション」 168 ページ
- 「CommLinks 接続パラメータ [LINKS]」 235 ページ
- 「DatabaseName 接続パラメータ [DBN]」 242 ページ
- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

次の例の DatabaseFile (DBF) 接続パラメータは、サンプル・データベース *demo.db* をロードして接続します。

```
DBF=samples-dir¥demo.db
```

samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

次の 2 つの例では、*cities.db* という名前のデータベース・ファイルをすでに起動していて、次のように Kitchener と名前変更したと仮定します。

```
dbeng10 cities.db -n Kitchener
```

データベースを起動して接続し、それを Kitchener と命名するには、次の手順に従います。

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


DBF=cities.db と指定すると、Kitchener という名前の実行中のデータベースへの接続が失敗します。

DatabaseKey 接続パラメータ [DBKEY]

暗号化されたデータベースを接続要求で起動します。

使用法

特に制限なし

値

文字列

デフォルト

なし

備考

接続要求で暗号化データベースを起動するときには、このパラメータを指定します。すでに実行している暗号化データベースに接続する場合は、このパラメータを指定する必要はありません。

暗号化キーは、大文字、小文字、数字、文字、特殊記号を含む文字列です。データベース・キーには、前後のスペースやセミコロンを含めることはできません。

クライアント・アプリケーションとデータベース・サーバ間の通信パケットを安全化するには、-ec サーバ・オプションとトランスポート・レイヤ・セキュリティを使用します。「トランスポート・レイヤ・セキュリティ」 949 ページを参照してください。

参照

- 「トランスポート・レイヤ・セキュリティを使用するクライアント・アプリケーションの設定」 963 ページ
- 「-ec サーバ・オプション」 161 ページ
- 「-ek データベース・オプション」 222 ページ
- 「-ep サーバ・オプション」 164 ページ
- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ
- 「Encryption 接続パラメータ [ENC]」 247 ページ

例

次のフラグメントは、DatabaseKey (DBKEY) 接続パラメータの使用方法を示します。

```
"UID=DBA;PWD=sql;ENG=myeng;DBKEY=V3moj3952B;DBF=samples-dir¥demo.db"
```

DatabaseName 接続パラメータ [DBN]

すでに実行されているデータベースに接続するときに、接続する必要のある、ロードしたデータベースを識別します。

まだ実行していないデータベースに接続する場合は、DatabaseFile (DBF) パラメータを使用します。

使用法

実行中のローカル・データベースかネットワーク・サーバ

値

文字列

デフォルト

デフォルト設定なし

備考

データベースがサーバで起動するたびにデータベース名が割り当てられます。これは、管理者が -n オプションを使用して割り当てるか、またはサーバがファイル名から拡張子とパスを削除した形で割り当てます。

注意
データベースの命名には、DatabaseSwitches (DBS) 接続パラメータで -n オプションを使用するよりも、DatabaseName (DBN) 接続パラメータを使用することをおすすめします。

接続するデータベースがすでに実行中の場合は、データベース・ファイルではなくデータベース名を指定してください。

実行中のデータベース名と DatabaseName (DBN) パラメータで指定した名前が一致した場合のみ、接続が行われます。

注意
データベース・ファイルとデータベース名の両方を指定すると、指定した名前の実行中のデータベースに接続します (データベース・ファイルは無視されます)。接続に失敗すると、データベース・ファイルとデータベース名の両方を使用して、データベースが自動的に起動されます。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

cities.db という名前のデータベース・ファイルを起動して、その名前を Kitchener に変更するには、次のコマンドを使用できます。

```
dbeng10 cities.db -n Kitchener
```

上記のコマンドを実行したと仮定すると、次に示すコマンドを使用して Kitchener という名前の実行中のデータベースに接続できます。

```
DBN=Kitchener
```


代わりに、次のコマンドを使用して Kitchener という実行中のデータベースに接続することもできます。

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

ただし、次のように指定すると、データベース Kitchener への接続が失敗します。

```
DBF=cities.db
```

DatabaseSwitches 接続パラメータ [DBS]

データベース起動時に、データベースに指定のオプションを提供します。

使用法

データベースがロードされていないときに、サーバに接続します。この接続パラメータは、サーバがまだ稼働していない場合に、指定されたデータベースとオプションでサーバを自動的に起動します。

値

文字列

デフォルト

オプションはありません。

備考

現在実行していないデータベースに接続するときにのみ、DatabaseSwitches を使用してください。DatabaseFile で指定されたデータベースをサーバが起動するとき、サーバは指定された DatabaseSwitches を使用して、データベースの開始オプションを決定します。

このパラメータを使用して指定できるのはデータベース・オプションのみです。サーバ・オプションは StartLine 接続パラメータを使用して指定してください。

「データベース・オプション」 218 ページを参照してください。

注意
データベースの命名には、DatabaseSwitches (DBS) 接続パラメータで -n オプションを使用するよりも、DatabaseName (DBN) 接続パラメータを使用することをおすすめします。

参照

- 「SQL Anywhere データベース・サーバ」 138 ページ
- 「StartLine 接続パラメータ [START]」 262 ページ
- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

次のコマンドは、コマンド・プロンプトですべて 1 行に入力して実行します。このコマンドは、デフォルトのデータベース・サーバに接続し、データベース・ファイル *demo.db* (DatabaseFile

(DBF) 接続パラメータ) をロードし、そのファイルに my-db (DatabaseName (DBN) 接続パラメータ) と名前を付け、これを読み込み専用モード (-r オプション) で開始します。

```
dbisql -c "UID=DBA;PWD=sql;DBF=samples-dir¥demo.db;DBN=my-db;DBS=-r"
```

samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

DataSourceName 接続パラメータ [DSN]

ODBC ドライバ・マネージャまたは Embedded SQL ライブラリに対して、レジストリ内またはシステム情報ファイル (デフォルトで *.odbc.ini*) での ODBC データ・ソース情報の検索場所を指示します。

使用法

特に制限なし

値

文字列

デフォルト

デフォルト・データ・ソース名なし

備考

データ・ソース名だけを ODBC に送信するのは、ODBC アプリケーションの一般的な手法です。ODBC ドライバ・マネージャと ODBC ドライバは、接続パラメータの残りの部分を含んだデータ・ソースを探します。

SQL Anywhere では、Embedded SQL アプリケーションも ODBC データ・ソースを使用して接続パラメータを保管できます。

参照

- 「FileDataSourceName 接続パラメータ [FILEDSN]」 250 ページ
- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ
- 「UNIX での ODBC データ・ソースの使用」 81 ページ

例

次のパラメータは、データ・ソース名を使用します。

```
DSN=My Database
```

DisableMultiRowFetch 接続パラメータ [DMRF]

ネットワーク上での複数ロー・フェッチをオフにします。

使用法

特に制限なし

値

YES、NO

デフォルト

NO

備考

デフォルトでは、データベース・サーバが単純なフェッチ要求を受信すると、アプリケーションは追加のローを要求します。このパラメータを YES に設定すると、この動作を無効にできます。

「プロシージャとトリガでのカーソルの使用」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

DisableMultiRowFetch (DMRF) 接続パラメータを YES に設定するのと、prefetch データベース・オプションを Off に設定するのとは、同じ効果があります。

「ローのプリフェッチ」『SQL Anywhere サーバ - プログラミング』を参照してください。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

次の接続文字列フラグメントは、プリフェッチを回避します。

```
DMRF=YES
```

EncryptedPassword 接続パラメータ [ENP]

パスワードを指定し、データ・ソースに暗号形式で保管します。

使用法

特に制限なし

値

文字列

デフォルト

なし

備考

データ・ソースは、ファイルかレジストリとしてディスクに保管されます。ディスクにパスワードを保管すると、セキュリティ上の問題が生じる場合があります。そのため、パスワードをデータ・ソースに入力すると、パスワードは暗号化形式で保管されます。

UNIX では、この情報はシステム情報ファイル (デフォルト名は *.odbc.ini*) に保持されます。

システム情報ファイルがどのように検索されるかについては、「UNIX での ODBC データ・ソースの使用」 81 ページを参照してください。

Password (PWD) 接続パラメータと EncryptedPassword (ENP) 接続パラメータの両方を指定した場合、Password (PWD) が優先されます。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

Encryption 接続パラメータ [ENC]

トランスポート・レイヤ・セキュリティまたは単純暗号化を使用してクライアント・アプリケーションとサーバ間で送信されるパケットを暗号化します。

使用法

TLS の場合は TCP/IP のみ。

NONE または **SIMPLE** の場合は特に制限なし。

値

Encryption= { NONE
| SIMPLE
| TLS(TLS_TYPE=*cipher***;**
[FIPS={ Y | N };]
TRUSTED_CERTIFICATES=*public-certificate* **) }**

デフォルト

NONE

備考

このパラメータは、トランスポート・レイヤ・セキュリティまたは単純暗号化を使用してクライアント・アプリケーションとデータベース・サーバ間の通信を安全化する場合に使用します。「トランスポート・レイヤ・セキュリティ」 949 ページを参照してください。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

Encryption (ENC) 接続パラメータには、次の引数を指定できます。

- **NONE** 暗号化されていない通信パケットを受け入れます。
- **SIMPLE** すべてのプラットフォームと SQL Anywhere の以前のバージョンでサポートされる単純暗号化で暗号化された通信パケットを受け入れます。単純暗号化では、サーバ認証、強力

な楕円曲線暗号化、RSA 暗号化、トランスポート・レイヤ・セキュリティのその他の機能は提供されません。

データベース・サーバが単純暗号化を受け入れ、暗号化なしを受け入れない場合、暗号化を使用しない TDS 接続以外の接続では、単純暗号化が使用されます。

-ec SIMPLE を指定してデータベース・サーバを起動すると、サーバは単純暗号化を使用した接続だけを受け入れます。TLS 接続 (ECC、RSA、RSA FIPS) は失敗し、暗号化を要求しない接続では単純暗号化が使用されます。

-ec SIMPLE,TLS(TLS_TYPE=ECC;...) を指定してデータベース・サーバを起動すると、サーバは ECC TLS 暗号化*または*単純暗号化を使用する接続だけを受け入れます。RSA 接続と RSA FIPS 接続はいずれも失敗し、暗号化を要求しない接続では単純暗号化が使用されます。

- ***cipher*** RSA 暗号化の場合は **RSA** を指定し、ECC 暗号化の場合は **ECC** を指定します。FIPS 承認の RSA 暗号化の場合は、**TLS_TYPE=RSA;FIPS=Y** を指定します。RSA FIPS は別の承認ライブラリを使用しますが、SQL Anywhere 9.0.2 以降で RSA を指定しているサーバと互換性があります。

  cipher に指定する暗号化が、証明書を作成するときに使用した暗号化 （RSA または ECC) と一致しない場合、接続は失敗します。

- ***public-certificate*** 信用された証明書を 1 つ以上含むファイルのパスとファイル名を指定します。FIPS 承認の RSA 暗号化を使用している場合は、RSA を使用して証明書を生成する必要があります。

サーバ認証での証明書のフィールドの検証については、「証明書フィールドの確認」 964 ページを参照してください。

デジタル証明書の使用については、「デジタル証明書の作成」 955 ページを参照してください。

CONNECTION_PROPERTY システム関数を使用して、現在の接続の暗号化設定値を取得できます。

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

接続によって使用されている暗号化のタイプに応じて、この関数は None、Simple、ecc_tls、rsa_tls、または rsa_tls_fips の 5 つの値のうちいずれか 1 つを返します。

「CONNECTION_PROPERTY 関数 [システム]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

参照

- 「トランスポート・レイヤ・セキュリティを使用するクライアント・アプリケーションの設定」 963 ページ
- 「-ec サーバ・オプション」 161 ページ
- 「-ek データベース・オプション」 222 ページ
- 「-ep サーバ・オプション」 164 ページ
- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ
- 「DatabaseKey 接続パラメータ [DBKEY]」 242 ページ

例

次の接続文字列フラグメントは、トランスポート・レイヤ・セキュリティと楕円曲線暗号化を使用して、TCP/IP リンクを通じて demo という名前のデータベース・サーバに接続します。

```
"ENG=demo;LINKS=tcpip;ENCRYPTION=tls(tls_type=ecc;trusted_certificates=sample.crt)"
```

次の接続文字列フラグメントは、トランスポート・レイヤ・セキュリティと RSA 暗号化を使用して、TCP/IP リンクを通じて demo という名前のデータベース・サーバに接続します。

```
"ENG=demo;LINKS=tcpip;ENCRYPTION=tls(tls_type=rsa;fips=n;trusted_certificates=rsaserver.crt)"
```

次の接続文字列フラグメントは、単純暗号化を使用し、TCP/IP リンクを通じて demo という名前のデータベース・サーバに接続します。

```
"ENG=demo;LINKS=tcpip;ENCRYPTION=simple"
```

EngineName 接続パラメータ [ENG]

接続する実行中のデータベース・サーバ名を指定します。これは ServerName の同意語です。

使用法

ネットワーク・サーバまたはパーソナル・サーバ

値

文字列

デフォルト

デフォルトのローカル・データベース・サーバ

備考

デフォルトのローカル・データベース・サーバに接続する場合は、EngineName は必要ありません。

複数のローカル・データベース・サーバが実行中、またはネットワーク・サーバに接続するときは、EngineName を指定する必要があります。[接続] ダイアログや ODBC アドミニストレータでは、[サーバ名] フィールドになります。

サーバを自動的に起動する場合、このパラメータを使ってサーバ名を指定できます。

サーバ名は、クライアント・コンピュータの文字セットに従って解釈されます。サーバ名に非 ASCII 文字を使用することは推奨できません。

この名前は、有効な識別子であることが必要です。ロング・サーバ・ネームは、プロトコルごとに異なる長さにトランケートされます。

プロトコル	トランケーションの長さ
TCP/IP	250 バイト
共有メモリ	250 バイト

プロトコル	トランケーションの長さ
SPX	32 バイト

Windows と UNIX では、データベース・サーバがバージョン 10.0.0 以降で、名前が次の長さを超えている場合、バージョン 9.0.2 以前のクライアントから接続することはできません。

- Windows 共有メモリの場合は、40 バイト
- UNIX 共有メモリの場合は、31 バイト
- TCP/IP の場合は、40 バイト

注意
配備されたアプリケーションの接続文字列には、 EngineName パラメータを含めることをおすすめします。これにより、コンピュータで複数の SQL Anywhere データベース・サーバが実行されている場合に、アプリケーションが確実に正しいサーバに接続できるため、タイミングに依存する接続エラーを防ぐことができます。

参照

- 「識別子」 『SQL Anywhere サーバ - SQL リファレンス』
- 「-n サーバ・オプション」 181 ページ
- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

サーバ Guelph に接続します。

```
ENG=Guelph
```

FileDataSourceName 接続パラメータ [FILEDSN]

接続するデータベースに関する情報を ODBC ファイル・データ・ソースが保有していることをクライアント・ライブラリに知らせます。

使用法

特に制限なし

値

文字列

デフォルト

デフォルト名なし

備考

ファイル・データ・ソースは、レジストリに保管される ODBC データ・ソースと同じ情報を持ちます。ファイル・データ・ソースは、簡単にエンド・ユーザに配布できるので、接続情報を各コンピュータ上で再構成する必要はありません。

ODBC と Embedded SQL アプリケーションのいずれも、ファイル・データ・ソースを使用できます。

参照

- 「DataSourceName 接続パラメータ [DSN]」 245 ページ
- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

ForceStart 接続パラメータ [FORCE]

サーバに接続せずにデータベース・サーバを起動します。

使用法

db_start_engine 関数と併用する場合のみ

値

YES、NO

デフォルト

NO

備考

ForceStart を YES に設定する場合、db_start_engine 関数は、すでに動作中のサーバがあってもサーバに接続せずにサーバを起動します。

参照

- 「db_start_engine 関数」『SQL Anywhere サーバ - プログラミング』
- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

Idle 接続パラメータ

接続のアイドル・タイムアウト時間を指定します。

使用法

TDS 接続や共有メモリ接続以外。他には特に制限なし。共有メモリ接続と TDS 接続 (jConnect を含む) では、SQL Anywhere Idle (IDLE) 接続パラメータは無視されます。

値の範囲

整数

デフォルト

なし

備考

Idle (IDLE) 接続パラメータは、現在の接続に対してのみ適用されます。同一サーバ上の複数の接続に異なるタイムアウト値を設定できます。

接続アイドル・タイムアウト値が設定されていないと、アイドル・タイムアウト値はサーバ側の設定によって制御されます。デフォルトは 240 分です。タイムアウト値が競合した場合、指定されているかいないかに関係なく、接続タイムアウト値がサーバ・タイムアウト値より優先されます。

IDLE 接続パラメータの最小値は 1 分で、サポートされている最大値は 32767 分です。0 に指定すると、接続に対するアイドル・タイムアウトのチェックがオフになります。

参照

- 「-ti サーバ・オプション」 197 ページ
- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

次の接続文字列フラグメントは、この接続のタイムアウト値を 10 分に設定します。

```
"ENG=myeng;LINKS=tcpip;IDLE=10"
```

Integrated 接続パラメータ [INT]

統合化ログインを試行できるかどうかを指定します。

使用法

特に制限なし

値

YES、**NO**

デフォルト

NO

備考

Integrated (INT) 接続パラメータには次の設定があります。

- **YES**　統合化ログインを行います。接続の試行が失敗し、login_mode オプションが Standard,Integrated に設定されている場合、標準ログインを試行します。
- **NO**　これはデフォルト設定です。統合化ログインは試行されません。

統合化ログインを使用するクライアント・アプリケーションでは、login_mode データベース・オプションを Integrated に設定してサーバを実行してください。

参照

- 「login_mode オプション [データベース]」 459 ページ
- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

次のデータ・ソース・フラグメントは、統合化ログインを使用します。

```
INT=YES
```

Kerberos 接続パラメータ [KRB]

データベース・サーバへの接続時に Kerberos 認証を使用できるかどうかを指定します。

使用法

Windows CE と NetWare を除くすべてのプラットフォーム

値

YES、**NO**、**SSPI**、*GSS-API-library-file* のいずれか

デフォルト

NO

備考

Kerberos [KRB] 接続パラメータには次の設定があります。

- **YES** Kerberos 認証ログインが試行されます。
- **NO** Kerberos 認証ログインは試行されません。これはデフォルトです。
- **SSPI** Kerberos 認証ログインが試行され、GSS-API ライブラリの代わりに組み込みの Windows SSPI インタフェースが使用されます。SSPI は Windows プラットフォームでのみ使用できます。また、Domain Controller Active Directory KDC 以外のキー配布センター (KDC) で SSPI を使用することはできません。Windows クライアント・コンピュータがすでに Windows ドメインにログインされている場合は、Kerberos クライアントをインストールまたは設定せずに、SSPI を使用できます。
- **GSS-API-library-file** Kerberos 認証ログインが試行されます。この文字列は、Kerberos GSS-API ライブラリ (または UNIX の共有オブジェクト) のファイル名を指定します。これは、Kerberos クライアントでデフォルトと異なる Kerberos GSS-API ライブラリ・ファイル名が使用されているか、コンピュータに複数の GSS-API ライブラリがインストールされている場合にだけ必要です。

Kerberos 認証ログインを使用している場合、UserID および Password 接続パラメータは無視されます。

Kerberos 認証を使用するには、Kerberos クライアントがインストールおよび設定され (SSPI の場合は不要)、ユーザが Kerberos にログインされ (有効な TGT がある)、データベース・サーバで Kerberos 認証ログインが有効になり設定されている必要があります。

参照

- 「-kl サーバ・オプション」 178 ページ
- 「-kr サーバ・オプション」 178 ページ
- 「-krb サーバ・オプション」 179 ページ
- 「Kerberos 認証の使用」 108 ページ
- 「GRANT 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「Windows で Kerberos ログインに SSPI を使用する」 114 ページ

例

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

Language 接続パラメータ [LANG]

接続の言語を指定します。

使用法

特に制限なし

値の範囲

2 文字の組み合わせで言語を表します。たとえば、LANG=DE の場合、デフォルト言語をドイツ語に設定します。

デフォルト

SALANG 環境変数、dblang ユーティリティ、インストーラの順に指定された言語

備考

この接続パラメータは接続用の言語を設定するものです。サーバが指定された言語をサポートしている場合に、サーバからのエラーや警告が指定された言語で配信されます。

言語を指定しない場合は、デフォルトの言語が使用されます。デフォルトの言語は、SALANG 環境変数、dblang ユーティリティ、インストーラの順で指定された言語です。

言語コードの詳細については、「ロケール言語の知識」 346 ページを参照してください。

この接続パラメータは接続のみに影響します。SQL Anywhere ツールとユーティリティから返されるメッセージはデフォルトの言語で表示されますが、サーバから戻されるメッセージは接続の言語で表示されます。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

LazyClose 接続パラメータ [LCLOSE]

CLOSE *cursor-name* データベース要求がキューイングされ、次のデータベース要求とともにサーバに送信されます。これで、カーソルを閉じるたびにネットワーク要求が行われることはなくなります。

使用法

特に制限なし

値

YES、**NO**

デフォルト

NO

備考

このパラメータが有効になっていると、実際にカーソルが閉じるのは次のデータベース要求が行われたときになります。CLOSE *cursor-name* データベース要求がキューイングされている間は、独立性レベル 1 のすべてのカーソル安定性ロックがカーソルに適用されます。

次の場合は、このオプションを有効にするとパフォーマンスが向上します。

- ネットワークの遅延時間が長い
- アプリケーションがカーソルを開く要求と閉じる要求を多数送信する

CLOSE *cursor-name* データベース要求の後、その次の要求をキャンセルすると、クライアント側でカーソルが閉じているように見える状態になることがまれにありますが、サーバ側では実際には閉じていません。その後で、同じ名前の別のカーソルを開こうとすると失敗します。アプリケーションが要求を頻繁にキャンセルする場合には、LazyClose の使用はおすすめしません。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

LivenessTimeout 接続パラメータ [LTO]

不完全な接続の終了を制御します。

使用法

ネットワーク・サーバのみ

非スレッド化 UNIX アプリケーションを除くすべてのプラットフォーム

値の範囲

整数 (秒)

デフォルト

なし

LivenessTimeout 値が設定されていない場合、LivenessTimeout はサーバ設定で制御されます。デフォルトは 120 秒です。

備考

接続が維持されていることを確認するため、クライアント／サーバの TCP/IP または SPX 通信プロトコルを介して、定期的に「**活性パケット**」が送信されます。活性要求や応答パケットを検出することなく、指定した LivenessTimeout 期間にわたってクライアントが実行されていると、通信は切断されます。

LivenessTimeout 値の 3 分の 1 から 3 分の 2 の期間で接続がパケットを送信しない場合に、活性パケットが送信されます。

サーバで 200 以上の接続がある場合に、サーバは指定された LivenessTimeout 値に基づいて LivenessTimeout 値が高いものを自動的に算出します。これにより、サーバはより多くの接続を効率よく処理できます。

別の方法として、[ODBC の設定] ダイアログの [ネットワーク] タブにある [活性タイムアウト] テキスト・ボックスに値を入力して、このパラメータを設定することもできます。

LivenessTimeout 接続パラメータの最小値は 30 秒で、最大値は 32767 秒です。0 に指定すると、接続に対する活性タイムアウトのチェックがオフになります。最小値より小さい 0 以外の値は最小値に再設定されます。たとえば、"LivenessTimeout=5" が含まれている接続文字列を指定すると、"LivenessTimeout=30" が使用されます。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

次の接続文字列フラグメントでは、LivenessTimeout 値を 10 分に設定します。

```
LTO=600
```

LogFile 接続パラメータ [LOG]

ファイルにクライアント・エラー・メッセージとデバッグ・メッセージを送信します。

使用法

特に制限なし

値

文字列

デフォルト

ログ・ファイルはありません。

備考

ファイルにクライアント・エラー・メッセージとデバッグ・メッセージを保存する場合、LogFile (LOG) 接続パラメータを使用します。

ファイル名にパスがない場合、クライアント・アプリケーションの現在の作業ディレクトリを基準にします。

LogFile (LOG) 接続パラメータは接続ごとに固有であるため、単一のアプリケーションで、接続ごとに異なる LogFile 引数を設定できます。

一般的なログ・ファイルの内容は次のとおりです。

```
Mon Aug 28 2006 12:29:46
12:29:46 Attempting to connect using:
UID=DBA;PWD=********;DBF='C:¥Documents and Settings¥All Users¥Documents¥SQL Anywhere 10
¥Samples¥demo.db';
ENG=demo10;START='C:¥Program Files¥SQL Anywhere 10¥win32¥dbeng10.exe';CON='Sybase
Central 1';
ASTOP=YES;LOG=c:¥mylog.txt
12:29:46 Attempting to connect to a running server...
12:29:46 Trying to start SharedMemory link ...

12:29:46 SharedMemory link started successfully

12:29:46 Attempting SharedMemory connection (no sasrv.ini cached address)

12:29:46 Failed to connect over SharedMemory

12:29:46 No server found, attempting to run START line...
12:29:47 Autostarted server, attempting to connect using:
UID=DBA;PWD=********;DBF='C:¥Documents and Settings¥All Users¥Documents¥SQL Anywhere 10
¥Samples¥demo.db';
ENG=demo10;START='C:¥Program Files¥SQL Anywhere 10¥win32¥dbeng10.exe';CON='Sybase
Central 1';ASTOP=YES
12:29:47 Attempting SharedMemory connection (no sasrv.ini cached address)

12:29:47 Connected to server over SharedMemory

12:29:47 Connected to SQL Anywhere Server version 10.0.0.2456
12:29:47 Application information:
12:29:47 IP=10.25.99.227;HOST=mymachine-XP;OS='Windows XP Build 2600 Service Pack
2';PID=0x21c;
THREAD=0xa38;EXE='C:¥Program Files¥SQL Anywhere 10¥Sybase Central 5.0.0¥win32¥scjview.exe';
VERSION=10.0.0.2456;API=iAnywhereJDBC;TIMEZONEADJUSTMENT=-240
12:29:47 Connected to the server, attempting to connect to a running database...
12:29:48 [    1] Connected to database successfully
12:29:53 [    1] The number of prefetch rows has been reduced to 168 due to the prefetch buffer
12:29:53 [    1] limit. Consider using the PrefetchBuffer connection parameter.
```

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

次のコマンド・ラインは、LogFile (LOG) 接続パラメータを使用して、サンプル・データベースに接続された Interactive SQL を起動します。

```
dbisql -c "DSN=SQL Anywhere 10 Demo;LOG=d:¥logs¥test.txt"
```


Password 接続パラメータ [PWD]

接続時のパスワードを指定します。

使用法

特に制限なし

値

文字列

デフォルト

パスワードの指定なし

備考

すべてのデータベース・ユーザにはパスワードがあります。パスワードをユーザに提供し、データベースへの接続が許可されるようにしてください。パスワードは最大長が 255 バイトで、大文字と小文字が区別されます。パスワードには、前後のスペースやセミコロンを含めることができます。

Password (PWD) 接続パラメータは暗号化されていません。データ・ソースにパスワードを保管する場合、EncryptedPassword (ENP) 接続パラメータを使用してください。Sybase Central と SQL Anywhere ODBC 設定ツールのいずれも、暗号化したパスワードを使用します。

Password (PWD) 接続パラメータと EncryptedPassword (ENP) 接続パラメータの両方を指定した場合、Password (PWD) 接続パラメータが優先されます。

別の方法として、[接続] ダイアログと [ODBC アドミニストレータ] ダイアログの [パスワード] テキスト・ボックスで、このパラメータを設定できます。

参照

- 「EncryptedPassword 接続パラメータ [ENP]」 246 ページ
- 「GRANT 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「大文字と小文字の区別」 『SQL Anywhere サーバ - SQL の使用法』
- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

次の接続文字列フラグメントは、ユーザ ID DBA とパスワード sql を指定します。

```
UID=DBA;PWD=sql
```

PrefetchBuffer 接続パラメータ [PBUF]

ローをバッファするためのメモリの最大容量を、バイト単位で設定します。

使用法

特に制限なし

値

整数 [**k** | **m**] デフォルト値と 8 MB の間

デフォルト

65536 (Windows CE を除くすべてのプラットフォーム)

16384 (Windows CE)

備考

PrefetchBuffer (PBUF) 接続パラメータは、プリフェッチされたローを格納するためにクライアントで割り付けられるメモリを制御します。デフォルト値はバイト単位ですが、**k** または **m** を使用してキロバイトまたはメガバイトの単位を指定できます。状況によっては、クライアントによってデータベース・サーバからプリフェッチされるローの数を増やすと、クエリのパフォーマンスが向上することがあります。プリフェッチされるローの数は、PrefetchRows (PROWS) と PrefetchBuffer (PBUF) 接続パラメータを使用して増やすことができます。

PrefetchBuffer (PBUF) 接続パラメータを増やすと、GET DATA 要求のバッファに使用できるメモリ容量も増えます。多数の GET DATA (SQLGetData) 要求を処理するアプリケーションでは、このように設定するとパフォーマンスが向上します。

以前のバージョンとの互換性を保つため、16384 未満の値が指定されるとキロバイト単位だと解釈されます。PrefetchBuffer 接続パラメータにおいて、k サフィックスを使用しないキロバイト単位での指定は使用されなくなりました。将来のバージョンでもサポートされません。「PrefetchRows 接続パラメータ [PROWS]」 260 ページを参照してください。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

次の接続文字列フラグメントを使用して、PrefetchBuffer によるメモリ制限によってプリフェッチされるローの数が減っているかどうかを判断できます。

```
...PrefetchRows=100;LogFile=c:¥client.txt
```

次の文字列を使用して、メモリ制限を 256 KB に増やすことができます。

```
...PrefetchRows=100;PrefetchBuffer=256k
```

PrefetchOnOpen 接続パラメータ

パラメータが有効であるときに、カーソルを開く要求を含むプリフェッチ要求が送信されます。

使用法

ODBC

値

YES、**NO**

デフォルト

NO

備考

このオプションを有効にすると、カーソルを開く要求を含むプリフェッチ要求が送信されます。これにより、カーソルを開くたびにローをフェッチするネットワーク要求は行われなくなります。カーソルを開くときにプリフェッチを実行するには、カラムをバインドしておきます。PrefetchOnOpen を使用するときに、カーソルを開いた後で最初のフェッチの前にカラムを再バインドすると、パフォーマンスが低下することがあります。

結果セットを返すクエリまたはストアド・プロシージャで ODBC の SQLExecute または SQLExecDirect を呼び出すと、カーソルが開きます。

次の場合は、このオプションを有効にするとパフォーマンスが向上します。

- ネットワークの遅延時間が長い
- アプリケーションがカーソルを開く要求と閉じる要求を多数送信する

PrefetchRows 接続パラメータ [PROWS]

データベースのクエリ時にプリフェッチされるローの最大数を設定します。

使用法

特に制限なし

値

整数

デフォルト

10

ADO.NET の場合は 200

備考

クライアントによってプリフェッチされるデータベース・サーバのローの数を増やすと、シングル・ロー・フェッチまたはワイド・フェッチで、0 または 1 の相対フェッチのみを行うカーソルのパフォーマンスを向上させることができます。ワイド・フェッチには、Embedded SQL 配列フェッチと ODBC ブロック・フェッチが含まれます。

特に次のような場合に、パフォーマンスが向上します。

- アプリケーションが非常に少ない絶対フェッチで多数の (何百もの) ローをフェッチする場合。
- アプリケーションがローをフェッチする頻度が高く、クライアントとサーバが同一コンピュータで動作しているか、高速ネットワークで接続されている場合。

- クライアント／サーバ通信にダイヤルアップ・リンクや広域ネットワークなどの伝送速度の遅いネットワークを使用している場合。

プリフェッチされるローの数は、PrefetchRows (PROWS) 接続パラメータと PrefetchBuffer (PBUF) 接続パラメータの両方によって制限されており、そのため、プリフェッチされたローの格納に使用できるメモリが制限されます。「PrefetchBuffer 接続パラメータ [PBUF]」 258 ページを参照してください。

プリフェッチできるロー数の最大値は 1000 です。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

次の接続文字列フラグメントは、プリフェッチされるローの数を 100 に設定します。

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

RetryConnectionTimeout 接続パラメータ [RetryConnTO]

クライアント・ライブラリ (dblib、ODBC、ADO など) に、サーバが見つからないうちは指定した時間が経過するまで接続を試行し続けるよう指示します。

使用法

特に制限なし

値

整数

デフォルト

0

備考

この接続パラメータで指定される値は、秒単位のタイムアウトです。接続試行のリトライ回数を示すカウンタ値では*ありません*。デフォルト値の 0 は、接続を 1 度だけ試行することを示します。間隔は 0.5 秒で、接続しようとしてデータベース・サーバが見つからなかった場合にだけリトライされます。それ以外のエラーは、すぐに返されます。データベース・サーバが見つからなかった場合、少なくとも RetryConnectionTimeout 接続パラメータに指定されている期間、接続が試行されます。

デフォルトの TCP タイムアウトは 5 秒で、接続文字列に指定が 5 秒未満の RetryConnTO が含まれている (LINKS=tcp;RetryConnTO=3 など) 場合にも、5 秒間、接続が試行されます。

参照

- 「Timeout プロトコル・オプション [TO]」 286 ページ

例

次の接続文字列フラグメントは、接続試行を最低 5 秒間リトライし続けるようクライアント・ライブラリに指示します。

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

ServerName 接続パラメータ [ENG]

これは EngineName (ENG) 接続パラメータと同じです。「EngineName 接続パラメータ [ENG]」 249 ページを参照してください。

StartLine 接続パラメータ [START]

アプリケーションからパーソナル・データベース・サーバを起動します。

使用法

組み込みデータベース

値

文字列

デフォルト

StartLine パラメータはありません。

備考

現在実行中でないデータベース・サーバに接続するときにかぎり、StartLine (START) 接続パラメータを指定します。StartLine 接続パラメータは、パーソナル・データベース・サーバを起動するコマンド・ラインです。CommLinks [LINKS] パラメータに TCPIP または SPX が含まれる場合、データベース・サーバは自動的には起動されません。

注意
データベース名、データベース・ファイル、サーバを指定する場合は、StartLine 接続パラメータではなく、DBN、DBF、ENG 接続パラメータを使用することをおすすめします。
次のコマンドでは、推奨される構文が使用されています。

```
START=dbeng10 -c 8M;ENG=mydb;DBN=mydb;DBF=c:¥sample.db
```

次の構文はおすすめしません。

```
START=dbeng10 -c 8M -n mydb "c:¥sample.db"
```

使用可能なオプションの詳細については、「SQL Anywhere データベース・サーバ」 138 ページを参照してください。

注意
StartLine 接続パラメータは、データベース・サーバを起動するために使用します。このパラメータを使用できるのは、指定したデータベース・サーバに接続できない場合、またはすでに実行されているデータベース・サーバ上でデータベースの起動および接続を行うことができない場合だけです。たとえば、次のように、データベースを実行するデータベース・サーバを起動するとします。

```
dbeng10 c:¥mydb.db
```

別のデータベースに接続します (ENG 接続パラメータを使用したデータベース・サーバ名の指定は行いません)。

```
dbisql -c "START=dbsrv10 -c 8M;DBN=seconddb;DBF=c:¥myseconddb.db;UID=DBA;PWD=sql"
```

この場合、dbsrv10 データベース・サーバは起動しません。その代わりに、*mydb.db* の起動に使用された dbeng10 データベース・サーバが、*myseconddb.db* を起動して接続するために使用されます。
ただし、ENG=*server-name* が指定されている場合で、*server-name* という名前のサーバがまだ実行されていないときは、dbsrv10 データベース・サーバが起動されます。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ
- 「CommLinks 接続パラメータ [LINKS]」 235 ページ

例

次のデータ・ソース・フラグメントは、8 MB のキャッシュでパーソナル・データベース・サーバを起動します。

```
StartLine=dbeng10 -c 8M;DBF=samples-dir¥demo.db
```

samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

Unconditional 接続パラメータ [UNC]

データベース・サーバへの接続があるときでも、db_stop_engine 関数を使用してデータベース・サーバを停止します。または、db_stop_database 関数を使用してデータベースを停止します。

使用法

db_stop_engine 関数と db_stop_database 関数のみ

値

YES、**NO**

デフォルト

NO

備考

db_stop_engine 関数は、データベース・サーバを停止します。db_stop_database 関数は、データベースを停止します。接続文字列で UNC=YES を指定すると、データベース・サーバまたはデータベースはアクティブな接続があるときでも停止されます。Unconditional が YES に設定されていない場合は、アクティブな接続がないときのみデータベース・サーバまたはデータベースが停止されます。

参照

- 「db_stop_database 関数」『SQL Anywhere サーバ - プログラミング』
- 「db_stop_engine 関数」『SQL Anywhere サーバ - プログラミング』
- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

Userid 接続パラメータ [UID]

データベースへのログインに使用するユーザ ID を指定します。

使用法

特に制限なし

値

文字列

デフォルト

なし

備考

データベースに接続するときは、統合化ログインまたは Kerberos ログインを使用している場合を除き、必ずユーザ ID を指定します。

参照

- 「接続パラメータの働き」 62 ページ
- 「接続パラメータのヒント」 86 ページ

例

次の接続文字列フラグメントは、ユーザ ID DBA とパスワード sql を指定します。

```
UID=DBA;PWD=sql
```

ネットワーク・プロトコル・オプション

ネットワーク・プロトコル・オプション (クライアントとサーバ両方のための) を使用して、さまざまなネットワーク・プロトコルの問題に対処できます。

ネットワーク・プロトコル・オプションは、サーバ・コマンドに指定できます。次に例を示します。

```
dbsrv10 -x tcpip(PARM1=value1;PARM2=value2;. . .),SPX
```

クライアント側では、CommLinks (LINKS) 接続パラメータとしてプロトコル・オプションを入力できます。

```
CommLinks=tcpip(PARM1=value1;PARM2=value2;. . .),SPX
```

パラメータにスペースがある場合、ネットワーク・プロトコル・オプションを二重引用符で囲むことにより、システム・コマンド・インタプリタによって適切に解析されます。

```
dbsrv10 -x "tcpip(PARM1=value 1;PARM2=value 2;...),SPX"
CommLinks="tcpip(PARM1=value 1;PARM2=value 2;...),SPX"
```

UNIX では、セミコロンがコマンドの区切り文字として解釈されるため、複数のパラメータを指定する場合にも二重引用符が必要です。

ブール・パラメータは、YES、Y、ON、TRUE、T、1 のいずれかによってオンになり、NO、N、OFF、FALSE、F、0 のいずれかによってオフになります。パラメータは、大文字と小文字を区別しません。

上記の例では、コマンドをすべて 1 行に入力しています。コマンドを設定ファイルに記述し、サーバ・オプションの @ を使って設定ファイルを呼び出すこともできます。

TCP/IP、HTTP、HTTPS、SPX プロトコル・オプション

TCP/IP、HTTP、HTTPS、および SPX で現在使用できるオプションを次に示します。

TCP/IP	HTTP と HTTPS	SPX
Broadcast [BCAST]	Certificate	BroadcastListener [BLISTENER]
BroadcastListener [BLISTENER]	Certificate_Password	DLL
ClientPort [CPORT]	DatabaseName [DBN]	DoBroadcast [DOBROAD]
DLL	KeepaliveTimeout [KTO]	ExtendedName [ENAME]
DoBroadcast [DOBROAD]	LocalOnly [LOCAL]	Host [IP]
Host [IP]	LogFile [LOG]	RegisterBindery [REGBIN]
LocalOnly [LOCAL]	LogMaxSize [LSize]	SearchBindery [BINSEARCH]

TCP/IP	HTTP と HTTPS	SPX
LDAP [LDAP]	LogOptions [LOpt]	Timeout [TO]
MyIP [ME]	LogFormat [LF]	
ReceiveBufferSize [RCVBUFSZ]	MaxConnections [MaxConn]	
SendBufferSize [SNDBUFSZ]	MaxRequestSize [MaxSize]	
ServerPort [PORT]	MyIP [ME]	
TDS	ServerPort [PORT]	
Timeout [TO]	Timeout [TO]	
VerifyServerName [VERIFY]		

Broadcast プロトコル・オプション [BCAST]

ブロードキャスト・メッセージの送信に使用する IP アドレスを指定します。

使用法

TCP/IP

値

文字列 (IP アドレス形式)

デフォルト

同一サブネット上のすべてのアドレスにブロードキャストします。

備考

デフォルトのブロードキャスト・アドレスは、ローカル IP アドレスとサブネット・マスクを使用して作成されます。サブネット・マスクは、IP アドレスのどの部分がネットワークを指定し、どの部分がホストを指定するかを示します。

たとえば、マスクが 255.255.255.0 のサブネット 10.24.98.x の場合、デフォルトのブロードキャスト・アドレスは 10.24.98.255 になります。

Windows プラットフォームで IPv6 アドレスを指定する場合、インタフェース識別子を使用する必要があります。UNIX プラットフォームでは、IPv6 アドレスのインタフェース識別子とインタフェース名の両方がサポートされます。Linux (カーネル 2.6.13 以上) では、インタフェース識別子が必要です。「SQL Anywhere での IPv6 サポート」 123 ページを参照してください。

例

次の接続文字列の例は、IPv6 を使用する場合にインタフェース番号 2 だけでブロードキャストするようクライアントに指示します。


```
LINKS=tcpip(BROADCAST=ff02::1%2)
```

BroadcastListener プロトコル・オプション [BLISTENER]

指定したポートのブロードキャスト受信を制御します。

使用法

SPX、TCP/IP (サーバ側)

値

YES、**NO**

デフォルト

YES

備考

このオプションを使用して、このポートのブロードキャスト受信を OFF に設定できます。

-sb 0 を指定することは、TCP/IP と SPX で BroadcastListener=NO に設定するのと同じことです。

参照

- 「-sb サーバ・オプション」 191 ページ

例

TCP/IP 接続と SPX 接続の両方に対応しますが、HOST プロトコル・オプションを使用した TCP/IP 接続を必要とするデータベース・サーバを起動します。

```
dbsrv10 -x tcpip(BroadcastListener=NO),spx ...
```

次に示すのは、データベース・サーバに接続するための接続文字列フラグメントです。

```
...LINKS=tcpip;HOST=myserver;...
```

Certificate プロトコル・オプション

暗号化証明書の名前を指定します。

使用法

HTTPS

値

文字列

デフォルト

デフォルトの証明書名なし

備考

この必須オプションにより、暗号化された証明書の名前を指定します。この証明書のパスワードは、Certificate_Password パラメータで指定してください。

例

特定の暗号化された証明書を使用するために、Web 接続が必要なサーバを起動します。

```
dbsrv10 -xs https(Certificate=cert.file;Certificate_Password=secret) ...
```

Certificate_Password プロトコル・オプション

暗号化証明書のパスワードを指定します。

使用法

HTTPS

値

文字列

デフォルト

デフォルトの証明書パスワードなし

備考

この必須オプションにより、Certificate パラメータで指定した暗号化された証明書に対応するパスワードを指定します。

例

特定の暗号化された証明書を使用するために、Web 接続が必要なサーバを起動します。

```
dbsrv10 -xs https(Certificate=cert.file;Certificate_Password=secret) ...
```

ClientPort プロトコル・オプション [CPORT]

TCP/IP を使用してクライアント・アプリケーションが通信するポート番号を指定します。

使用法

TCP/IP (クライアント側のみ)

値

整数

デフォルト

ネットワークの実装によって、接続ごとに動的に割り当てられます。ファイアウォールの制限がない場合は、このパラメータを使用しないようおすすめします。

備考

このオプションは、ファイアウォールを介した接続のために提供されています。ファイアウォール・ソフトウェアは、TCP/UDP ポートに従ってフィルタします。ファイアウォールの理由によって必要な場合以外は、このパラメータを使用しないようおすすめします。

ClientPort オプションは、クライアント・アプリケーションが TCP/IP を使って通信するポート番号を指定します。単一のポート番号、または個々のポート番号の組み合わせやポート番号の範囲を指定できます。次に例を示します。

- (cport=1234)
- (cport=1234,1235,1239)
- (cport=1234-1238)
- (cport=1234-1237,1239,1242)

指定されたデータ・ソースや接続文字列を使用して複数の接続を確立する場合、ポート番号のリストや範囲を指定することをおすすめします。ポート番号を 1 つだけ指定すると、アプリケーションが維持できるのは、一度に 1 つの接続のみとなります。また、1 つの接続を閉じた後は、数分のタイムアウト時間が生じます。その間、指定されたポートを使って新しい接続は作成できません。ポート番号のリストや範囲を指定すると、アプリケーションは、いずれかのポート番号との接続が確立するまで、試行を続けます。

参照

- 「Host プロトコル・オプション [IP]」 272 ページ
- 「DoBroadcast プロトコル・オプション [DOBROAD]」 271 ページ
- 「ServerPort プロトコル・オプション [PORT]」 283 ページ
- 「ファイアウォール経由の接続」 124 ページ

例

次の接続文字列フラグメントは、ポート 6000 を使用するアプリケーションから、ポート 5000 を使用する my-server という名前のサーバへの接続を確立します。

```
CommLinks=tcpip(ClientPort=6000;ServerPort=5000);ServerName=my-server
```

次の接続文字列フラグメントは、ポート 5050 ー 5060、5040、5070 を使用できるアプリケーションから、デフォルトのサーバ・ポートを使用する my-server という名前のサーバへ通信する接続を確立します。

```
CommLinks=tcpip(ClientPort=5040,5050-5060,5070);
ServerName=my-server
```

DatabaseName プロトコル・オプション [DBN]

Web 要求を処理するときに使用するデータベース名を指定します。また、REQUIRED や AUTO キーワードを使用して URI の一部としてデータベース名が必要かどうかを指定します。

使用法

HTTP、HTTPS

値の範囲

AUTO、**REQUIRED**、*データベース名*

デフォルト

AUTO

備考

このパラメータが REQUIRED に設定されている場合は、URI がデータベース名を指定します。

このパラメータが AUTO に設定されている場合は、URI がデータベース名を指定できますが、必須ではありません。URI にデータベース名が含まれていない場合は、サーバでのデフォルトのデータベースを Web 要求の処理に使用します。AUTO に設定されている場合、サーバは URI にデータベース名が含まれているかどうかを推測しなければならないため、あいまいにならないように Web サイトを設計してください。

このパラメータにデータベースが設定されている場合は、このデータベースを使用してすべての Web 要求を処理します。URI にはデータベース名を含めないでください。

例

次のコマンドは 2 つのデータベースを起動しますが、HTTP 経由でのアクセスを許可されているのはそのうちの 1 つだけです。

```
dbsrv10 -xs http(DBN=web) samples-dir¥demo.db web.db
```

DLL プロトコル・オプション

クライアントが使用する必要のある DLL バージョンを指定します。

使用法

TCP/IP、SPX (Windows XP/200x/Vista) クライアント側のみ

値

文字列

デフォルト

Windows XP/200x/Vista のクライアントでは、デフォルトは *ws2_32.dll* (Winsock 2.2) です。

備考

このオプションは、必要なネットワーク・インタフェース機能がデフォルトのプロトコル・スタックとは異なる DLL に含まれるような、未テストの TCP/IP プロトコル・スタックをサポートするために使用されます。クライアントは、要求された機能を名前付き DLL で検索します。

Windows CE のクライアントでは、サポートされているバージョンは *wsock32.dll* (Winsock 1.1) だけです。

例

Windows XP での次のコマンドは、Winsock 2.2 を使用します。

```
dbping -c "ENG=my-eng;LINKS=tcpip(DLL=ws2_32.dll)"
```

DoBroadcast プロトコル・オプション [DOBROAD]

クライアントがデータベース・サーバを検索する方法と、データベース・サーバが起動時にブロードキャストを実行するかどうかを制御します。

使用法

TCP/IP、SPX

値

ALL、**NONE**、**DIRECT** (クライアント側)

YES、**NO** (サーバ側)

デフォルト

ALL (クライアント側)

YES (サーバ側)

備考

クライアントでの使用法　DoBroadcast=ALL に設定した場合、ブロードキャストを実行してデータベース・サーバを検索します。最初は、ローカル・サブネットにブロードキャストされます。HOST= を指定した場合、各ホストにはブロードキャスト・パケットも送信されます。TCP/IP の場合、ブロードキャスト・パケットはすべて UDP パケットです。SPX の場合、バインダリにサーバがないときにかぎり、ブロードキャストを実行します。SPX の場合、ブロードキャスト・パケットはすべて IPX パケットです。

DoBroadcast=DIRECT を設定した場合、データベース・サーバを検索するときに、ローカル・サブネットへのブロードキャストは実行されません。ブロードキャスト・パケットは、HOST (IP) プロトコル・オプション・リストにあるホストにのみ送信されます。DoBroadcast=DIRECT を指定する場合は、HOST (IP) プロトコル・オプションが必要です。

DoBroadcast=NONE を指定すると、UDP または IPX ブロードキャストは使用されず、サーバ・アドレスキャッシュ (*sasrv.ini*) は無視されます。指定した HOST/PORT との TCP/IP 接続または SPX 接続が直接行われ、サーバ名が検証されます。TCP/IP の場合は、VerifyServerName (VERIFY) プロトコル・オプションを NO に設定して、サーバ名を検証しないようにすることもできます。HOST (IP) プロトコル・オプションは必須のパラメータですが、LDAP を使用する場合を除き、ServerPort (PORT) プロトコル・オプションは省略可能です。

DIRECT と NONE の場合は、HOST オプションでサーバ・ホストを指定します。

サーバでの使用法　DoBroadcast=NO に設定すると、起動時にデータベース・サーバがブロードキャストを実行して、同じ名前の他のサーバを検索しないようにできます。この設定が役に立つ場合もまれにありますが、通常は必要ありません。

例

次のコマンドは、ブロードキャストを実行してデータベース・サーバを検索することなく、クライアントを起動します。サーバは silver という名前のコンピュータ上でのみ検索されます。

```
CommLinks=tcpip(DOBROADCAST=DIRECT;HOST=silver) demo
```

ExtendedName プロトコル・オプション [ENAME]

SPX のデータベース・サーバ名における不正な文字列を指定できます。

使用法

SPX (Windows XP/200x/Vista 以外のプラットフォーム)

値

YES、**NO**

デフォルト

NO

備考

有効な SAP 名に対する Novell 標準により、次の文字は使用できません。

```
¥ / : ; , * ? + -
```

demo-1 という名前が付けられたサーバを起動する場合、デフォルトの動作はハイフン (-) を削除して demo1 というサーバを起動しようとします。ExtendedName をオンにすることにより、名前がそのまま使用されます。

これは SPX ポート・パラメータであり、サーバを起動する場合にのみ役立ちます。

警告
このオプションは SAP 標準に反するので、使用する場合は注意してください。

例

次のコマンドは、NetWare サーバ demo-1 を起動します。

```
load dbsrv10.nlm -x spx(ExtendedName=YES) demo-1
```

Host プロトコル・オプション [IP]

クライアント・ライブラリの検索対象となる、直接接続されているネットワークの外部にある追加コンピュータを指定します。

使用法

TCP/IP、SPX

値

文字列

デフォルト

追加コンピュータはありません。

備考

HOST は、クライアント・ライブラリの検索対象となる、直接接続されているネットワークの外部にある追加コンピュータを指定します。サーバ上では、重複する名前のサーバが起動するのを回避するように検索されます。

TCP/IP の場合、アドレスには *hostname* の IP アドレスを使用できます。オプションで、PORT 値を指定することもできます。

Windows プラットフォームで IPv6 アドレスを指定する場合、インタフェース識別子を使用する必要があります。UNIX プラットフォームでは、IPv6 アドレスのインタフェース識別子とインタフェース名の両方がサポートされます。Linux (カーネル 2.6.13 以上) では、インタフェース識別子が必要です。「SQL Anywhere での IPv6 サポート」 123 ページを参照してください。

SPX の場合は、*a*:*b*:*c*:*d*:*e*:*f*/*g*:*h*:*i*:*j* という形式のアドレスを使用します。ここで、*a*:*b*:*c*:*d*:*e*:*f* はサーバのノード番号 (Ethernet カード・アドレス) であり、*g*:*h*:*i*:*j* はネットワーク番号です。

-z オプションを使用すると、サーバの起動時にサーバ・メッセージ・ウィンドウにアドレス情報が表示されます。また、LogFile が指定されている場合、アプリケーションはこの情報をログ・ファイルに書き込みます。

カンマで区切ったアドレスのリストを使って、複数のコンピュータを検索できます。また、コロンを区切り文字として使用してポート番号を IP アドレスに追加できます。別の方法として、HOST=myhost;PORT=5000 のように、ホストとサーバ・ポートを明示的に指定することもできます。IPv6 アドレスの場合は、(fe80::5445:5245:444f):2638 のように、アドレスをカッコで囲む必要があります。

1 つのパラメータに複数の値を指定するには、カンマで区切ったリストを使用します。複数のポートとサーバを指定する場合には、PORT パラメータではなく、HOST (IP) プロトコル・オプションにポートを指定することで、特定のポートと特定のサーバを関連付けることができます。

TCP/IP を使用する場合、IP と HOST は同義語です。SPX の場合は、HOST を使用します。

参照

- 「ClientPort プロトコル・オプション [CPORT]」 268 ページ

例

次の接続文字列フラグメントは、kangaroo と 197.75.209.222 (ポート 2369) というコンピュータを検索し、データベース・サーバを見つけるようにクライアントに指示します。

```
LINKS=tcpip(IP=kangaroo,197.75.209.222:2369)
```

次の接続文字列フラグメントは、my-server と kangaroo というコンピュータを検索し、データベース・サーバを見つけるようにクライアントに指示します。ポート 2639 で実行中の最初に応答したホストへの接続が試行されます。


```
LINKS=tcpip(HOST=my-server,kangaroo;PORT=2639)
```

次の接続文字列フラグメントは、ポート 1234 で稼働する host1 上のサーバと、ポート 4567 で稼働する host2 上のサーバを検索するようにクライアントに指示します。クライアントは、ポート 4567 の host1 またはポート 1234 の host2 は検索しません。

```
LINKS=tcpip(HOST=host1:1234,host2:4567)
```

次の接続文字列フラグメントは、IPv6 アドレス上でサーバを探すようクライアントに指示します。

```
LINKS=tcpip(HOST=fe80::5445:5245:444f)
```

Host プロトコル・オプションを指定した IPv6 アドレスの使用例を次に示します。

```
Global scope address, unique everywhere, so no interface index is required
-c "links=tcpip(Host=fd77:55d:59d9:56a:202:55ff:fe76:df19)" // no index required
-c "links=tcpip(Host=fd77:55d:59d9:56a:202:55ff:fe76:df19%2)" // all communication is done through
interface 2
-c "links=tcpip(Host=fd77:55d:59d9:56a:202:55ff:fe76:df19%eth0)" // all communication is done through
eth0

Link scope address, addresses are unique on each interface
-c "links=tcpip(Host=fe80::202:55ff:fe76:df19)" // possibly ambiguous (this host may exist through both
eth0 and eth1)
-c "links=tcpip(Host=fe80::202:55ff:fe76:df19%2)" // not ambiguous because it must use interface 2
-c "links=tcpip(Host=fe80::202:55ff:fe76:df19%eth0)" // not ambiguous because it must use eth0
```

KeepaliveTimeout プロトコル・オプション [KTO]

データベース・サーバが要求の完了まで待機する最大時間 (秒単位) を指定します。

使用法

HTTP

値

整数

デフォルト

60

備考

通常は、接続は要求の終了ごとに閉じられます。クライアントが Keep-Alive オプションを要求した場合は、要求と応答が終了しても HTTP 接続が開いたままになるので、複数の要求を同じ接続で実行できます。

接続が開かれると、クライアントは指定された回数、完全な HTTP 要求 (POST 要求の本文を含む) を送ります。Keep-Alive が要求された接続では、結果の送信後にタイムアウトがリセットされるので、各要求の開始時は新しい接続を開いているような状況になります。

接続がタイムアウトされないようにする場合は、kto=0 を指定します。

KeepaliveTimeout と Timeout プロトコル・オプションの違いは、KeepaliveTimeout が接続を開いた時点からの合計時間を指定するのに対し、Timeout が要求内でパケット間の最大時間を指定することです。

参照

- 「HTTP ヘッダの使用」『SQL Anywhere サーバ - プログラミング』
- 「Timeout プロトコル・オプション [TO]」 286 ページ

LDAP プロトコル・オプション [LDAP]

クライアントは、IP アドレスを指定せずにデータベース・サーバを検索できます。

使用法

TCP/IP

値

YES、**NO**、*ファイル名*

デフォルト

ON

デフォルトのファイル名は *saldap.ini* です。

備考

データベース・サーバ自体を LDAP サーバに登録することにより、クライアントが LDAP サーバにクエリを実行できます。これにより、WAN 上で動作するクライアントやファイアウォールを経由するクライアントが IP アドレスを指定せずにサーバを検索できます。また、検出ユーティリティ (dblocate) もそのようなサーバを検索できるようになります。

LDAP=*filename* と指定すると、LDAP のサポートがオンになり、指定したファイルが設定ファイルとして使用されます。LDAP=YES と指定すると、LDAP のサポートがオンになり、*saldap.ini* が設定ファイルとして使用されます。

ファイル非表示ユーティリティを使用すると、単純暗号化によって *saldap.ini* ファイルの内容を非表示できます。「.ini ファイルの内容の非表示」 657 ページを参照してください。

LDAP は TCP/IP でのみ使用されます。

参照

- 「LDAP サーバを使用した接続」 126 ページ

LocalOnly プロトコル・オプション [LOCAL]

クライアントがローカル・コンピュータ上のサーバ (存在する場合) のみに接続できるようにします。

使用法

TCP/IP、HTTP、HTTPS

値

YES、**NO**

デフォルト

NO

備考

サーバ名が一致するサーバが、ローカル・コンピュータで見つからない場合、サーバは自動的に起動しません。

LocalOnly (LOCAL) プロトコル・オプションは、DoBroadcast=ALL (デフォルト) の場合にのみ役立ちます。

LocalOnly=YES を設定すると、サーバから他のコンピュータへのブロードキャスト応答が無視された場合を除き、通常のブロードキャスト・メカニズムが使用されます。

LocalOnly (LOCAL) プロトコル・オプションをサーバで使用して、ローカル・コンピュータへの接続に限定させることができます。リモート・コンピュータからの接続試行ではこのサーバは検索されず、検出 [dblocate] ユーティリティからはこのサーバは見えません。LocalOnly (LOCAL) プロトコル・オプションを YES に設定してサーバを稼働すると、接続や CPU の制限を受けずにネットワーク・サーバがパーソナル・サーバとして稼働します。

参照

- 「Broadcast プロトコル・オプション [BCAST]」 266 ページ

LogFile プロトコル・オプション [LOG]

データベース・サーバが Web 要求に関する情報を書き込むファイル名を指定します。

使用法

HTTP、HTTPS

値の範囲

ファイル名

デフォルト

なし

備考

データベース・サーバが Web 要求に関する情報を書き込むファイル名を指定します。

参照

- 「LogFormat プロトコル・オプション [LF]」 277 ページ

- 「LogMaxSize プロトコル・オプション [LSIZE]」 278 ページ
- 「LogOptions プロトコル・オプション [LOPT]」 278 ページ

LogFormat プロトコル・オプション [LF]

ログ・ファイルに書き込まれるメッセージのフォーマットと、表示されるフィールドを制御します。

使用法

HTTP、HTTPS

値の範囲

フォーマット文字列

デフォルト

@T - @W - @I - @P - "@M @U @V" - @R - @L - @E

備考

このパラメータは、ログ・ファイルに書き込まれるメッセージのフォーマットと、表示されるフィールドを制御します。文字列に表示される場合、各メッセージが書き込まれると現在の値が次のコードに置き換えられます。

- **@@** @ 文字
- **@B** 要求の処理が開始された日付／時刻 (エラーにより要求をキューイングできない場合を除く)
- **@C** クライアントが接続した日付／時刻
- **@D** 要求に関連するデータベース名
- **@E** エラーが発生した場合の、エラー・メッセージ・テキスト
- **@F** 要求の処理が終了した日付／時刻
- **@I** クライアントの IP アドレス
- **@L** ヘッダと本文を含んだ応答の長さ (バイト)
- **@M** HTTP 要求方式
- **@P** 要求に関連するリスナ・ポート
- **@Q** 要求の処理がキューイングされた日付／時刻 (エラーにより要求をキューイングできない場合を除く)
- **@R** HTTP 応答のステータス・コードおよび説明
- **@S** HTTP ステータス・コード
- **@T** 現在のログ・エントリが書き込まれた日付／時刻

- **@U**　要求 URI
- **@V**　要求 HTTP バージョン
- **@W**　要求を処理した時間 (@F − @B)、またはエラーにより要求が処理されなかった場合は 0.000

参照

- 「LogFile プロトコル・オプション [LOG]」 276 ページ
- 「LogMaxSize プロトコル・オプション [LSIZE]」 278 ページ
- 「LogOptions プロトコル・オプション [LOPT]」 278 ページ

LogMaxSize プロトコル・オプション [LSIZE]

トランザクション・ログ・ファイルの最大サイズを制御します。

使用法

HTTP、HTTPS

値

整数 [**k** | **m** | **g**]

デフォルト

0

備考

ログ・ファイルが指定したサイズに達すると、名前が変更されて、別のログ・ファイルが作成されます。LogMaxSize が 0 の場合、ログ・ファイルのサイズは無制限です。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ **k**、**m**、**g** を使用します。

参照

- 「LogFile プロトコル・オプション [LOG]」 276 ページ
- 「LogFormat プロトコル・オプション [LF]」 277 ページ
- 「LogOptions プロトコル・オプション [LOPT]」 278 ページ

LogOptions プロトコル・オプション [LOPT]

トランザクション・ログに記録されるメッセージ・タイプを指定します。

使用法

HTTP、HTTPS

値の範囲

NONE、**OK**、**INFO**、**ERRORS**、**ALL**、*ステータス・コード*、**REQHDRS**、**RESHDRS**、**HEADERS**

デフォルト

ALL

備考

使用可能な値には、特定のメッセージ・タイプと HTTP ステータス・コードを選択するキーワードがあります。カンマで区切って複数の値を指定できます。

次のキーワードは、ログするメッセージのカテゴリを制御します。

- **NONE** 何もログしない
- **OK** ログ要求が正しく完了 (20x HTTP ステータス・コード)
- **INFO** 終了または未変更ステータス・コードを返すログ要求 (30x HTTP ステータス・コード)
- **ERRORS** すべてのエラーをログ (40x と 50x HTTP ステータス・コード)
- **ALL** すべての要求をログ

次の共通 HTTP ステータス・コードも使用可能です。特定のステータス・コードを返す要求をログするために使用できます。

- **C200** OK
- **C400** 不正な要求
- **C401** 無認可
- **C403** 禁止
- **C404** 見つからない
- **C408** 要求タイムアウト
- **C501** 未実装
- **C503** サービス利用不可

加えて、次のキーワードを使用してログされたメッセージの詳細情報を取得できます。

- **REQHDRS** 要求のロギング時に、ログ・ファイルに要求ヘッダも書き込みます。
- **RESHDRS** 要求のロギング時に、ログ・ファイルに応答ヘッダも書き込みます。
- **HEADERS** 要求のロギング時に、ログ・ファイルに要求ヘッダと応答ヘッダの両方を書き込みます (REQHDRS、RESHDRS と同様)。

参照

- 「LogFile プロトコル・オプション [LOG]」 276 ページ
- 「LogFormat プロトコル・オプション [LF]」 277 ページ
- 「LogMaxSize プロトコル・オプション [LSIZE]」 278 ページ

MaxConnections プロトコル・オプション [MAXCONN]

データベース・サーバで許可される同時接続の数を指定します。

使用法

HTTP、HTTPS

値の範囲

サイズ

デフォルト

5 (パーソナル・サーバ)

ライセンスされている接続数 (ネットワーク・サーバ)

備考

サーバで許可される同時接続の数。値 0 は、無制限であることを示します。

参照

- 「MaxRequestSize プロトコル・オプション [MAXSIZE]」 280 ページ

MaxRequestSize プロトコル・オプション [MAXSIZE]

データベース・サーバで許可できる最大要求サイズを指定します。

使用法

HTTP、HTTPS

値

整数 [**k** | **m** | **g**]

デフォルト

100k

備考

サーバで許可される最大要求サイズ。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ **k**、**m**、**g** を使用します。要求サイズがこれを超える場合は、接続が閉じられて 413 ENTITY TOO LARGE 応答がクライアントに戻されます。この値は要求サイズのみを制限するもので、応答サイズは制限しません。値 0 はこの制限を無効にしますが、細心の注意を払って使用してください。この制限がないと、悪質なクライアントがサーバに過負荷をかけたり、メモリ不足を引き起こしたりする可能性があります。

参照

- 「MaxConnections プロトコル・オプション [MAXCONN]」 280 ページ

例

次のコマンド・ライン (すべて 1 行に入力) は、150000 バイトまでのサイズの要求を受け入れるようサーバに指示します。

```
dbsrv10 -xs http{MaxRequestSize=150000}
```

MyIP プロトコル・オプション [ME]

クライアントがアドレス情報を決定するかどうかを制御します。

使用法

TCP/IP、HTTP、HTTPS

値

文字列

備考

MyIP (ME) プロトコル・オプションは、複数のネットワーク・アダプタを持つコンピュータに対して指定します。

各アダプタには IP アドレスがあります。デフォルトでは、データベース・サーバは検出したすべてのネットワーク・インタフェースを使用します。データベース・サーバがすべてのネットワーク・インタフェースで受信しないようにする場合、使用する各インタフェースのアドレスを MyIP (ME) プロトコル・オプションで指定します。

IP 番号としてキーワード NONE が指定されている場合は、アドレス情報を決定しようとしません。NONE キーワードは、複数のネットワーク・カードを持つコンピュータや、リモート・アクセス (RAS) ソフトウェアとネットワーク・カードを持つコンピュータなど、この操作により大きな負荷がかかるコンピュータ上のクライアントで使用するものです。サーバでは使用しないでください。

複数の IP アドレスを指定する場合はカンマで区切ります。

Windows プラットフォームで IPv6 アドレスを指定する場合、インタフェース識別子を使用する必要があります。UNIX プラットフォームでは、IPv6 アドレスのインタフェース識別子とインタフェース名の両方がサポートされます。Linux (カーネル 2.6.13 以上) では、インタフェース識別子が必要です。「SQL Anywhere での IPv6 サポート」 123 ページを参照してください。

例

次のコマンド・ライン (すべて 1 行に入力) は、サーバに 2 つのネットワーク・カードを使用することを指示します。

```
dbsrv10 -x tcpip(MyIP=192.75.209.12,192.75.209.32) "samples-dir¥demo.db"
```

次のコマンド・ライン (すべて 1 行に入力) は、IPv6 ネットワーク・カードを使用することをデータベース・サーバに指示します。

```
dbsrv10 -x tcpip(MyIP=fe80::5445:5245:444f) "samples-dir¥demo.db"
```

samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

次の接続文字列フラグメントは、クライアントがアドレス情報を決定しないように指定します。

```
LINKS=tcpip(MyIP=NONE)
```

ReceiveBufferSize プロトコル・オプション [RCVBUFSZ]

TCP/IP プロトコル・スタックが使用するバッファのサイズを設定します。

使用法

TCP/IP

値

整数 [**k** | **m** | **g**]

デフォルト

コンピュータによって異なります。

備考

ネットワークに対する BLOB のパフォーマンスが重要な場合は、この値を増加できます。デフォルトでは、バッファ・サイズはバイト単位で指定します。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ **k**、**m**、または **g** を使用してください。

RegisterBindery プロトコル・オプション [REGBIN]

SPX リンクをロードする場合、データベース・サーバはネットワーク上のアクティブなバインダリに名前を登録するかどうかを制御します。

使用法

SPX (サーバ側のみ)

値

YES、**NO**

デフォルト

YES

備考

SPX リンクをロードする場合、デフォルトでは、データベース・サーバはネットワーク上のアクティブなバインダリに名前を登録しようとします。名前を登録しないようにするには、RegisterBindery を NO に設定します。この場合、クライアント・ライブラリはパケットをブロードキャストすることによって、SPX を使用しているデータベース・サーバを探します。

SearchBindery プロトコル・オプション [BINSEARCH]

NetWare バインダリでデータベース・サーバが検索されるかどうかを制御します。

使用法

SPX

値

YES、**NO**

デフォルト

YES

備考

データベース・サーバは、通常、同名の既存データベース・サーバを探すときにバインダリを検索します。SEARCHBINDERY=NO を使用すると、データベース・サーバを探すときに NetWare バインダリは検索されません。

SendBufferSize プロトコル・オプション [SNDBUFSZ]

TCP/IP プロトコル・スタックが使用するバッファのサイズを設定します。

使用法

TCP/IP

値

整数 [**k** | **m** | **g**]

デフォルト

コンピュータによって異なります。

備考

単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ **k**、**m**、**g** を使用します。ネットワークに対する BLOB のパフォーマンスが重要な場合は、この値を増加できます。

ServerPort プロトコル・オプション [PORT]

データベース・サーバが実行されているポートを指定します。

使用法

TCP/IP、HTTP、HTTPS

値

整数

デフォルト

TCP/IP のデフォルト値は 2638、HTTP のデフォルト値は 80、HTTPS のデフォルト値は 443 です。

備考

Internet Assigned Numbers Authority は、SQL Anywhere データベース・サーバに対して、TCP/IP 通信に使用するためのポート番号 2638 を割り当てています。ただし、その他のアプリケーションはこの予約ポートの使用を許可されていないわけではないため、データベース・サーバと別のアプリケーション間でアドレスが衝突する可能性があります。

データベース・サーバの場合、ServerPort プロトコル・オプションは TCP/IP を使用する通信のポート番号を指定します。単一のポート番号、または個々のポート番号の組み合わせやポート番号の範囲を指定できます。次に例を示します。

- (port=1234)
- (port=1234,1235,1239)
- (port=1234-1238)
- (port=1234-1237,1239,1242)

ポート番号のリストや範囲を指定すると、アプリケーションは、いずれかのポート番号との接続が確立するまで、試行を続けます。

ネットワーク・プロトコル・オプションを使用して異なるポートを指定した場合でも、データベース・サーバはほとんどのオペレーティング・システムで常に UDP ポート 2638 を使って受信します。そのため、アプリケーションは、ポート番号を指定しなくてもデータベース・サーバに接続できます。

クライアントの場合、ServerPort プロトコル・オプションは、データベース・サーバが TCP/IP 通信を受信する 1 つまたは複数のポートをそのクライアントに知らせます。クライアントは、ServerPort (PORT) プロトコル・オプションで指定されたすべてのポートにブロードキャストして、サーバを検索します。

Web サーバを使用している場合、デフォルトでは、データベース・サーバは標準 HTTP ポートと HTTPS ポートをそれぞれ 80 と 443 で受信します。

TCP/IP ポート番号 2638 (デフォルト) を使用してデータベース・サーバを起動した場合は、サーバは UDP ポート 2638 でも受信します。データベース・サーバは UDP ポートで受信し、それらのポートで要求に応答するため、クライアントはサーバ名によってデータベース・サーバを検索できます。

データベース・サーバの TCP/IP ポート番号が 2638 ではない場合、サーバは同じ UDP を TCP/IP ポートとして受信します。

Mac OS X での相違点
Mac OS X では、同一の UDP ポートに複数のプロセスをバインドすることはできません。データベース・サーバがこれらのいずれかのプラットフォームで実行されている場合、指定の UDP ポート、またはポートの指定がないときはポート 2638 のみで受信します。
これは、サーバがデフォルト・ポート (2638) を使用しない場合は、クライアントが TCP/IP ポート番号を指定することが*必要*であることを意味します。
たとえば、データベース・サーバがコマンド dbsrv10 -n MyServer *samples-dir*/demo.db によって起動される場合、同じサブネット上のクライアントは接続パラメータ ENG=MyServer;LINKS=tcpip を使用してサーバを見つけることができます。また別のサーバが、コマンド dbsrv10 -n SecondServer -x tcpip(PORT=7777) *samples-dir*/demo.db を使用して Mac OS X で起動される場合は、同じサブネット上のクライアントは接続パラメータ ENG=SecondServer;LINKS=tcpip(PORT=7777) を使用してサーバを見つけることができます。
データベース・サーバが Mac OS X 以外のプラットフォームで実行された場合は、クライアントが PORT パラメータを指定する必要はありません。
さらに、Mac OS X では、SQL Anywhere データベース・サーバがポート 2638 をすでに使用していて、PORT プロトコル・オプションなしで 2 番目のネットワーク・データベース・サーバが起動された場合は、そのネットワーク・サーバの起動は失敗します。この理由は、ユーザはサーバのポート番号を知っている必要があり、それを接続パラメータで指定する必要があるからです。
パーソナル・サーバの接続には、通常共有メモリが使用されるため、パーソナル・サーバは、ポート 2638 が使用中であっても正常に起動します。

例

次の例は、PORT プロトコル・オプションを使用して、サーバの起動に使うポートを指定する方法を示しています。

1. ネットワーク・データベース・サーバを起動します。

   ```
   dbsrv10 -x tcpip -n server1
   ```

   ポート番号 2638 が取得されました。

2. 別のデータベース・サーバを起動しようとします。

   ```
   dbsrv10 -x tcpip -n server2
   ```

   デフォルトのポートは現在使用されているので、サーバは別のポートを起動します Mac OS X の場合は失敗します。

3. すでにコンピュータ上の別の Web サーバがポート 80 を使用しているか、またはこのポート番号でサーバを起動するためのパーミッションがない場合は、8080 などの代替ポートを受信するサーバを起動できます。

   ```
   dbsrv10 -xs http(port=8080) -n server3 web.db
   ```

TDS プロトコル・オプション

データベース・サーバへの TDS 接続が許可されるかどうかを制御します。

使用法

TCP/IP (サーバ側のみ)

値

YES、**NO**

デフォルト

YES

備考

データベース・サーバへの TDS 接続を許可しないためには、TDS を NO に設定します。サーバに暗号化された接続だけを許可したい場合、TDS 接続を許可しないようにする方法はこのプロトコル・オプションしかありません。

例

次のコマンドは、Open Client または jConnect アプリケーションからの接続は許可しないで、TCP/IP プロトコルを使ってデータベース・サーバを起動します。

```
dbsrv10 -x tcpip(TDS=NO) ...
```

Timeout プロトコル・オプション [TO]

通信確立時に、応答を待つ時間を秒単位で設定します。

使用法

TCP/IP、SPX、HTTP、HTTPS

値

整数 (秒)

デフォルト

TCP/IP の場合は 5

HTTP と HTTPS の場合は 30

備考

また、切断時に応答を待つ時間も指定します。TCP/IP または SPX 通信の確立が困難である場合は、より長い時間の設定が必要なことがあります。

データベース・サーバでは、同名のサーバを検索するためにブロードキャストを送信した後に待機する時間です。サーバの起動時にのみ使用され、クライアント接続には影響しません。

サーバで HTTP または HTTPS を使用する場合、このパラメータは要求受信時の最大許容アイドル時間を指定します。この制限に達した場合は、接続が閉じられて 408 REQUEST TIMEOUT 応答がクライアントに返されます。値 0 はアイドル・タイムアウトを無効にしますが、細心の注意を払って使用してください。この制限がないと、悪質なクライアントがサーバのリソースを消費したり、他のクライアントが接続できなくなる可能性があります。

例

次のデータ・ソース・フラグメントは、タイムアウト時間を 20 秒にして、TCP/IP 通信リンクのみを起動します。

```
...
CommLinks=tcpip(TO=20)
...
```

VerifyServerName プロトコル・オプション [VERIFY]

クライアントが接続前にデータベース・サーバ名を確認する必要があるかどうかを制御します。

使用法

TCP/IP (クライアント側のみ)

値

YES、**NO**

デフォルト

YES

備考

TCP を介して接続している場合、DoBroadcast=NONE パラメータを指定すると、クライアントによって TCP 接続が行われ、検出されたサーバと検索対象サーバの名前が一致しているかどうかが検証されます。VerifyServerName=NO を指定すると、サーバ名は検証されません。そのため、IP アドレスかポートさえわかっていれば、SQL Anywhere クライアントから SQL Anywhere サーバに接続できます。

この場合も接続文字列にサーバ名を指定する必要がありますが、その指定は無視されます。VerifyServerName (VERIFY) プロトコル・オプションは、DoBroadcast=NONE を指定した場合にのみ使用します。

サーバが -sb 0 または BroadcastListener=NO を使用している場合、クライアントはサーバに接続するために DoBroadcast=NONE を指定する必要はありません。ただし、HOST= を指定することは必要です。dblocate ユーティリティはサーバを検索しません。

> **注意**
> このパラメータは、各サーバにユニークなサーバ名を付けることができないなど、特別な場合にのみ使用してください。接続するときは、ユニークなサーバ名を使用することをおすすめします。サーバへの接続に最適な方法は、各サーバにユニークなサーバ名を付け、その名前を使用することです。

参照

- 「DoBroadcast プロトコル・オプション [DOBROAD]」 271 ページ

パート II. データベースの設定

パート II では、SQL Anywhere に必要なファイル、データベースの制限事項、データベース・プロパティとオプションの設定方法について説明します。ここでは、SQL Anywhere のインストール環境を設定して、国際言語の問題を処理する方法についても説明します。

第 7 章

データベース・ファイルの処理

目次

データベース・ファイルの概要

基本データベース・ファイル

各データベースには、次のような関連ファイルがあります。

- **データベース・ファイル** このファイルはデータベース情報を格納します。通常、このファイルの拡張子は *.db* です。「データベースの編集」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **トランザクション・ログ** このファイルはデータベースの変更記録を格納するもので、リカバリと同期に必要です。通常、このファイルの拡張子は *.log* です。「トランザクション・ログ」 816 ページを参照してください。
- **テンポラリ・ファイル**
  データベース・サーバはテンポラリ・ファイルを使用して、必要な情報をデータベース・セッション中に格納します。データベースが停止すると、データベース・サーバはたとえ実行中であっても、テンポラリ・ファイルを破棄します。サーバが生成した名前に拡張子 *.tmp* を付けたものが、テンポラリ・ファイル名です。

  テンポラリ・ファイルのロケーションは、データベース・サーバの起動時に -dt サーバ・オプションを使用して指定できます。テンポラリ・ファイルのロケーションを指定せずにデータベース・サーバを起動した場合は、以下の環境変数がこの順序のとおりにチェックされます。

  - SATMP 環境変数
  - TMP 環境変数
  - TMPDIR 環境変数
  - TEMP 環境変数

  これらの環境変数がいずれも定義されていない場合、テンポラリ・ファイルは、Windows オペレーティング・システムの場合は現在のディレクトリ、UNIX の場合は */tmp* ディレクトリに作成されます。

  テンポラリ・ファイルは、サーバが作成、管理、削除します。テンポラリ・ファイル用に使用できる、十分な空き領域があることだけ確認してください。テンポラリ・ファイルに使用できる領域に関する情報は、sa_disk_free_space プロシージャを使用すると取得できます。「sa_disk_free_space システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

事前定義の DB 領域

SQL Anywhere は、データベースに対して、次に挙げる事前定義の DB 領域を使用します。

DB 領域	名前
メイン・データベース・ファイル	SYSTEM

DB 領域	名前
テンポラリ・ファイル	TEMPORARY または TEMP
トランザクション・ログ・ファイル	TRANSLOG
トランザクション・ログ・ミラー	TRANSLOGMIRROR

これらの名前を使用して、ユーザ定義の DB 領域を作成することはできません。また、事前定義の DB 領域を削除することはできません。

前述の事前定義の DB 領域と同じ名前が付いたユーザ定義の DB 領域を使用して、10.0.0 以前のバージョンからアップグレードすると、これらの DB 領域を参照している SQL 文は、事前定義の DB 領域ではなく、ユーザ定義の DB 領域を参照していると見なされます。事前定義の DB 領域を参照するように設定するには、ユーザ定義の DB 領域を削除するか、ユーザ定義の DB 領域の名前を変更して事前定義の DB 領域とは別の名前を指定する方法しかありません。

ALTER DBSPACE 文は、事前定義の DB 領域の名前をサポートするため、これらに対して領域を追加できます。「ALTER DBSPACE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

DB_EXTENDED_PROPERTY 関数も、事前定義の DB 領域の名前を受け入れます。「DB_EXTENDED_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

その他のファイル

次のようなファイルもデータベース・システムの一部になります。

- **DB 領域ファイル** データベース・ファイルに加えて、データを複数の個別ファイルに分散できます。「CREATE DBSPACE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  DB 領域については、「追加 DB 領域の使用」 294 ページを参照してください。

- **トランザクション・ログ・ミラー・ファイル** データの保全のため、トランザクション・ログのミラー・コピーを作成できます。通常、このファイルの拡張子は *.mlg* です。「トランザクション・ログのメディア障害からの保護」 830 ページを参照してください。

追加 DB 領域の使用

通常は大容量データベース向け
ほとんどのデータベースでは、データベース・ファイルは 1 つだけで十分です。しかし、大容量データベースを使用していると、多くの場合、追加データベース・ファイルが必要になります。また、追加データベース・ファイルは、別々のファイルにある関連した情報をまとめる場合に便利なツールです。

データベースを初期化すると、データベースにはデータベース・ファイルが 1 つ含まれます。この最初のデータベース・ファイルを「**メイン・ファイル**」と呼びます。デフォルトでは、すべてのデータベース・オブジェクトとすべてのデータがこのメイン・ファイルに配置されます。

DB 領域は、データ用の領域をさらに作成する追加のデータベース・ファイルです。1 つのデータベースには 13 個までファイルを保管できます (初期ファイル 1 つと 12 の DB 領域)。各テーブルは、そのインデックスとともに、単一のデータベース・ファイルに含まれている必要があります。CREATE DBSPACE という SQL コマンドで、新しいファイルをデータベースに追加できます。

各データベース・ファイルの最大容量は、2^{28} (約 2 億 6800 万) データベース・ページです。たとえば、データベース・ページ・サイズが 4 KB のデータベース・ファイルが作成されると、そのファイルのサイズは 1 テラバイト (2^{28}*4 KB) まで増やすことができます。しかし実際には、ファイルが作成された物理ファイル・システムで許容される最大ファイル・サイズが、最大許容サイズに大きく影響します。

一般に使用されているファイル・システムの多くはファイルの最大サイズが 2 GB ですが、Windows が使用している NTFS ファイル・システムのように、データベース・ファイルを最大サイズまで利用できるものもあります。データベースにあるデータの量が最大ファイル・サイズを超える場合は、データを複数のデータベース・ファイルに分割する必要があります。また、関連オブジェクトをまとめる場合など、サイズ制限以外の理由で複数の DB 領域を作成する場合もあります。

サポートされるオペレーティング・システムごとのファイルの最大サイズについては、「SQL Anywhere のサイズと数の制限」 564 ページを参照してください。

sa_disk_free system システム・プロシージャを使用して、DB 領域に使用可能な領域に関する情報を取得できます。「sa_disk_free_space システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

SYSFILE システム・ビューには、TEMPORARY、TRANSLOG、および TRANSLOGMIRROR の DB 領域を除く、データベースのすべての DB 領域に関する情報が含まれています。「SYSFILE システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

既存のデータベースの分割

既存のデータベース・オブジェクトをいくつかの DB 領域に分割する場合は、データベースをアンロードし、生成済みのコマンド・ファイル (デフォルトでは *reload.sql* という名前のファイル) をデータベース再構築用に修正します。*reload.sql* ファイルで、メイン・ファイルに配置しないテーブルごとに、CREATE TABLE 文に IN 句を追加して DB 領域を指定します。

参照

- 「CREATE DBSPACE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「DB_EXTENDED_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』
- 「CREATE TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「UNLOAD TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』

DB 領域の作成

新しいデータベース・ファイル (「DB 領域」) は、Sybase Central から、または CREATE DBSPACE 文を使用して作成します。新しい DB 領域のためのデータベース・ファイルは、メイン・ファイルが存在するディスク・ドライブ、または別のディスク・ドライブに作成できます。DB 領域を作成するには DBA 権限が必要です。

各データベースについて、メイン DB 領域に加えて最大 12 の DB 領域を作成できます。オブジェクトを格納する DB 領域を指定するには、default_dbspace データベース・オプションを設定します。

DB 領域へのテーブル配置

新しく作成された DB 領域は空です。新しいテーブルを作成するときは、CREATE TABLE 文に IN 句を指定し、テーブルを特定の DB 領域に配置することもできれば、事前に default_dbspace オプションを設定しておくこともできます。IN 句を指定しなかった場合や default_dbspace オプションの設定を変更しなかった場合、テーブルはメイン DB 領域に配置されます。

各テーブルは、そのテーブルが作成された DB 領域にすべて格納されます。デフォルトでは、テーブルと同じ DB 領域にインデックスが配置されますが、CREATE INDEX 文の一部として IN 句を指定して別の DB 領域に配置することもできます。

♦ DB 領域を作成するには、次の手順に従います (Sybase Central の場合)。

1. データベースの [DB 領域] フォルダを開きます。
2. [ファイル] メニューから [新規] – [DB 領域] を選択します。

   [DB 領域作成] ウィザードが表示されます。
3. ウィザードの指示に従います。

   新しい DB 領域が [DB 領域] フォルダに表示されます。

♦ DB 領域を作成するには、次の手順に従います (SQL の場合)。

- CREATE DBSPACE 文を実行します。

例

次のコマンドを使用して、メイン・ファイルと同じディレクトリにあるファイル *library.db* 内に、library という名前の新しい DB 領域を作成します。

```
CREATE DBSPACE library
AS 'library.db'
```

次のコマンドを使用して、LibraryBooks テーブルを作成し、それを library DB 領域に配置します。

```
CREATE TABLE LibraryBooks (
title CHAR(100),
author CHAR(50),
isbn CHAR(30)
) IN library
```

次のコマンドを使用して、library という名前の新しい DB 領域を作成し、この DB 領域をデフォルト DB 領域に設定したうえで、この DB 領域に LibraryBooks テーブルを作成します。

```
CREATE DBSPACE library
AS 'e:¥¥dbfiles¥¥library.db';
SET OPTION default_dbspace = 'library';
CREATE TABLE LibraryBooks (
  title CHAR(100),
  author CHAR(50),
  isbn CHAR(30),
);
```

参照

- 「CREATE DBSPACE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「default_dbspace オプション [データベース]」 444 ページ
- 「テーブルの作成」『SQL Anywhere サーバ - SQL の使用法』
- 「CREATE INDEX 文」『SQL Anywhere サーバ - SQL リファレンス』

データベース・ファイル用領域の事前割り付け

新しいデータベース・ファイルを作成するときは、CREATE DATABASE 文の DATABASE SIZE 句を使用するか、dbinit -dbs オプションを指定することによって、データベース領域を事前に割り付けることができます。「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』と「初期化ユーティリティ (dbinit)」 662 ページを参照してください。

データベースを使用していると、必要に応じてデータベース・ファイルのサイズが自動的に増大します。データベース・ファイルを頻繁に更新していると、ディスク上のファイルが過度に断片化し、パフォーマンスが低下することがあります。サイズの小さな多数の領域を割り付けるには、サイズの大きい領域を 1 つ割り付けるよりも時間がかかります。変更の頻度が高いデータベースの場合は、Sybase Central または ALTER DBSPACE 文を使用して、DB 領域やトランザクション・ログに対し、ディスク領域を事前に割り付けることができます。

データベース・ファイルのプロパティを変更するには、DBA 権限が必要です。

パフォーマンス上のヒント
ディスク領域を事前に割り付けてからディスク断片化解除ユーティリティを実行すると、ディスク・ドライブのあちこちにデータベース・ファイルが断片化されるのを、確実に防ぐことができます。データベース・ファイルの断片化が進むと、パフォーマンスが低下します。

- **領域を事前に割り付けるには、次の手順に従います (Sybase Central の場合)。**

1. [DB 領域] フォルダを開きます。
2. 対象の DB 領域を右クリックし、ポップアップ・メニューから [領域の事前割り付け] を選択します。
3. DB 領域に追加する領域のサイズを入力します。領域は、ページ、バイト、キロバイト (KB)、メガバイト (MB)、ギガバイト (GB)、またはテラバイト (TB) 単位で追加できます。
4. [OK] をクリックします。

- **領域を事前に割り付けるには、次の手順に従います (SQL の場合)。**

1. データベースに接続します。
2. ALTER DBSPACE 文を実行します。

例

SYSTEM の DB 領域サイズを 200 ページ増やします。

```
ALTER DBSPACE system
ADD 200;
```

SYSTEM の DB 領域サイズを 400 メガバイト増やします。

```
ALTER DBSPACE system
ADD 400 MB;
```

参照

- 「DB 領域の作成」 295 ページ
- 「ALTER DBSPACE 文」 『SQL Anywhere サーバ - SQL リファレンス』

DB 領域の削除

DB 領域を削除するには、Sybase Central または DROP DBSPACE 文を使用します。DB 領域を削除する前に、その DB 領域を使用するテーブルとインデックスをすべて削除する必要があります。DB 領域を削除するには DBA 権限が必要です。

- **DB 領域を削除するには、次の手順に従います (Sybase Central の場合)。**

1. [DB 領域] フォルダを開きます。
2. 目的の DB 領域を右クリックし、ポップアップ・メニューから [削除] を選択します。

- **DB 領域を削除するには、次の手順に従います (SQL の場合)。**

1. データベースに接続します。
2. DROP DBSPACE 文を実行します。

参照

- 「テーブルの削除」『SQL Anywhere サーバ - SQL の使用法』
- 「DROP 文」『SQL Anywhere サーバ - SQL リファレンス』

ユーティリティ・データベースの使用

「**ユーティリティ・データベース**」は、物理的な実体を持たない幻データベースです。この機能によって、CREATE DATABASE などのデータベース・ファイル管理文を、既存の物理データベースに接続しなくても実行できます。ユーティリティ・データベースにはデータベース・ファイルがないため、データを入れることができません。

たとえば、ユーティリティ・データベースに接続してから次の文を実行すると、ディレクトリ *c:¥temp* にデータベース *new.db* が作成されます。

```
CREATE DATABASE 'c:¥¥temp¥¥new.db';
```

「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

ユーティリティ・データベースを使用して、接続プロパティとサーバ・プロパティの値を取り出すこともできます。

たとえば、ユーティリティ・データベースに対して次の文を実行すると、デフォルトの照合順が返され、作成するデータベースに使用できます。

```
SELECT PROPERTY( 'DefaultCollation' );
```

接続とサーバ プロパティの詳細については、「データベース・プロパティの概要」 518 ページを参照してください。

ユーティリティ・データベースに使用できる文

次に、ユーティリティ・データベースに接続するときに実行できる文を示します。

- ALTER DATABASE *dbfile* ALTER TRANSACTION LOG
- CREATE DATABASE
- CREATE DECRYPTED FILE
- CREATE ENCRYPTED FILE
- DROP DATABASE
- GRANT CONNECT TO DBA IDENTIFIED BY *new-password*
- RESTORE DATABASE
- REVOKE CONNECT FROM DBA
- START DATABASE
- STOP DATABASE
- STOP ENGINE
- SELECT (FROM 句または WHERE 句なし)

ユーティリティ・データベースへの接続

サーバへの接続時に utility_db をデータベース名として指定すると、データベース・サーバ上のユーティリティ・データベースを起動できます。-su サーバ・オプションを使用すると、DBA ユーザのユーティリティ・データベース・パスワードを設定したり、ユーティリティ・データベースへの接続を無効にしたりできます。-su オプションを指定しないでユーティリティ・データベースを起動すると、パーソナル・サーバとネットワーク・サーバのユーザ ID やパスワードの要件が変わります。

パーソナル・データベース・サーバの場合、-su オプションが指定されないと、ユーティリティ・データベースへの接続に関するセキュリティ上の制限はありません。パーソナル・サーバの場合、ユーザ ID として DBA を指定します。また、パスワードも指定する必要がありますが、どのようなパスワードでもかまいません。パーソナル・データベース・サーバに接続できるユーザであればファイル・システムに直接アクセスできることから、パスワードによるユーザの選別は行われません。

ユーティリティ・データベースのパスワードをプレーン・テキストで入力することを回避するには、-su オプションを使用してパスワードを含むファイルを作成し、dbfhide ユーティリティを使用してファイルの内容を読みにくくします。たとえば、ユーティリティ・データベースのパスワードを含むファイルの名前が *util_db_pwd.cfg* である場合を考えてみます。dbfhide を使用してこのファイルを読みにくくし、*util_db_pwd_hide.cfg* という名前に変更します。

```
dbfhide util_db_pwd.cfg util_db_pwd_hide.cfg
```

その後、*util_db_pwd_hide.cfg* ファイルを使用して、ユーティリティ・データベースのパスワードを指定できます。

```
dbsrv10 -su @util_db_pwd_hide.cfg -n my_server c:¥mydb.db
```

「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。

ネットワーク・サーバの場合、-su オプションが指定されないと、ユーザ ID として DBA を指定する必要があります。また、データベース・サーバの実行ファイルと同じディレクトリにある *util_db.ini* ファイルに格納されているパスワードも指定します。このディレクトリはサーバ上にあるため、ファイルへのアクセスを制御でき、したがって誰がパスワードを使用するかを制御できます。パスワードでは大文字と小文字が区別されます。

注意
util_db.ini ファイルの使用は推奨されません。-su サーバ・オプションを使用して、ユーティリティ・データベースの DBA ユーザのパスワードを指定してください。「-su サーバ・オプション」 196 ページを参照してください。

♦ パーソナル・サーバ上のユーティリティ・データベースに接続するには、次の手順に従います (Interactive SQL の場合)。

1. 次のコマンドを使用してデータベース・サーバを起動します。

   ```
   dbeng10.exe -n TestEng
   ```

   セキュリティを高めるため、-su オプションを使用して、ユーティリティ・データベース・パスワードを指定します。

2. Interactive SQL を起動します。
3. [接続] ダイアログで、ユーザ ID に **DBA** と入力し、ブランク以外のパスワードを入力します。パスワード自体は確認されませんが、空白にすることはできません。
4. [データベース] タブで、データベース名として **utility_db**、サーバ名として **TestEng** を入力します。

5. [OK] をクリックして接続します。

   Interactive SQL は、TestEng というパーソナル・サーバ上のユーティリティ・データベースに接続します。

♦ ネットワーク・サーバ上のユーティリティ・データベースに接続するには、次の手順に従います (Interactive SQL の場合)。

1. 次のコマンドを使用してデータベース・サーバを起動します。

   ```
   dbsrv10.exe -n TestEng -su 9Bx231K
   ```

2. Interactive SQL を起動します。
3. [接続] ダイアログで、ユーザ ID に **DBA** と入力し、-su オプションで指定されているパスワードを入力します。
4. [データベース] タブで、データベース名として **utility_db**、サーバ名として **TestEng** を入力します。
5. [OK] をクリックして接続します。

   Interactive SQL は、TestEng というネットワーク・サーバ上のユーティリティ・データベースに接続します。

「データベースへの接続」 59 ページと「-su サーバ・オプション」 196 ページを参照してください。

注意
ユーティリティ・データベースに接続するときに REVOKE CONNECT FROM DBA を実行すると、以降のユーティリティ・データベースへの接続が無効になります。これは、REVOKE CONNECT を実行する前に確立していた接続を使用するか、データベース・サーバを再起動しないかぎり、以降はユーティリティ・データベースに接続できないことを意味します。「REVOKE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

ネットワーク・データベース・サーバでの util_db.ini の使用 (旧式)

注意
util_db.ini ファイルの使用は推奨されないため、-su サーバ・オプションを使用して、ユーティリティ・データベースの DBA パスワードを指定することをおすすめします。

util_db.ini の使用は、データベース・サーバを実行するコンピュータの物理的なセキュリティに依存します。これは、*util_db.ini* ファイルがテキスト・エディタで簡単に読むことができるからです。

ネットワーク・サーバの場合、デフォルトでは、-su オプションまたは *util_db.ini* を使用しないでユーティリティ・データベースに接続することはできません。*util_db.ini* を使用する場合、ファイルにパスワードが格納されています。このファイルは、データベース・サーバの実行プログラムと同じディレクトリに配置され、次のテキストを含んでいます。


```
[UTILITY_DB]
PWD=password
```

不用意な直接アクセスから *util_db.ini* ファイルの内容を保護するには、ファイル非表示ユーティリティ (dbfhide) を使って、単純暗号化をファイルに追加します。オペレーティング・システムの機能を使用して、サーバのファイル・システムへのアクセスを制限することもできます。

ini ファイルの難読化の詳細については、「.ini ファイルの内容の非表示」 657 ページを参照してください。

ファイル管理文の EXECUTE パーミッション

データベース・サーバのオプション -gu は、誰がファイル管理文を実行できるかを制御します。このオプションを使用して、特定の管理タスクを実行できるユーザを指定します。

ファイル管理文を使用するためのパーミッションには、4 つのレベルがあります。

-gu オプション	影響	適用対象
all	誰でもファイル管理文を実行できる	ユーティリティ・データベースを含むすべてのデータベース
none	誰もファイル管理文を実行できない	ユーティリティ・データベースを含むすべてのデータベース
DBA	DBA 権限を持つユーザだけがファイル管理文を実行できる	ユーティリティ・データベースを含むすべてのデータベース
utility_db	ユーティリティ・データベースに接続できるユーザだけがファイル管理文を実行できる	ユーティリティ・データベースのみ

「-gu サーバ・オプション」 176 ページを参照してください。

例

ファイル管理文の使用を禁じるには、-gu オプションの none パーミッション・レベルを使用してデータベース・サーバを起動します。次のコマンドを入力すると、データベース・サーバが起動され、サーバ名が TestSrv になります。*mytestdb.db* データベースがロードされますが、そのサーバを使用してデータベースを作成または削除したり、他のファイル管理文を実行したりすることはできません。これはリソース作成権の有無や、ユーティリティ・データベースをロードして接続できるかどうかには関係ありません。

```
dbsrv10.exe -n TestSrv -gu none c:¥mytestdb.db
```

ユーティリティ・データベースのパスワードを知っているユーザだけにファイル管理文の実行を許可するには、コマンド・プロンプトで次のコマンドを入力してサーバを起動します。

```
dbsrv10 -n TestSrv -su secret -gu utility_db
```

次のコマンドを使用することによって、Interactive SQL をクライアント・アプリケーションとして起動し、サーバ TestSrv に接続して、ユーティリティ・データベースをロードし、ユーザを接続します。

```
dbisql -c "UID=DBA;PWD=secret;DBN=utility_db;ENG=TestSrv"
```

上記のコマンドを正常に実行できた場合、ユーザはユーティリティ・データベースに接続し、ファイル管理文を実行できるようになります。

第 8 章

SQL Anywhere の環境変数

目次

SQL Anywhere 環境変数の概要

SQL Anywhere では、環境変数にさまざまな情報を登録しています。設定しなければならない環境変数は、状況によって異なります。

SQL Anywhere サーバの場合、特定のサーバに設定された環境変数を参照するには、サーバを起動するときに -ze オプションを指定します。「-ze オプション」 210 ページを参照してください。

Windows での環境変数の設定

SQL Anywhere のインストーラによって、コンピュータのプロパティに環境変数 PATH、SQLANY10、SQLANYSH10 が作成されます。既存の場合は、変更されます。SQL Anywhere をインストールしたら、コンピュータを再起動して、これらの環境変数を反映する必要があります。

その他の環境変数は、コンピュータのプロパティを変更して設定できます。また、SET コマンドを使用してコマンド・プロンプトやバッチ・ファイルから変更することも可能です。

Mac OS X の Finder の環境変数の設定

SQL Anywhere のインストーラによって、環境変数 DYLD_LIBRARY_PATH、ODBCINI、PATH、SQLANY10、SQLANYSH10 が設定されます。SQL Anywhere をインストールしたら、ログアウトしてからもう一度ログインして、環境変数の設定を反映させる必要があります。再起動する必要はありません。

SQL Anywhere アプリケーションを Finder から使用するユーザごとに、dbmodenv ユーティリティを実行して、GUI 環境 (*~/.MacOSX/environment.plist*) を設定する必要があります。これらの設定を有効にするには、各ユーザはログアウトしてから、もう一度ログインする必要があります。

dbmodenv ユーティリティは */Applications/SQLAnywhere10/System/bin32* にあります。

ターミナル・セッションは、Finder の環境変数を継承しません。ここでは、ターミナル・セッションの環境変数を設定する手順を説明します。

UNIX と Mac OS X での環境変数の設定

SQL Anywhere 10 のインストール後、各ユーザはシステムの環境変数を設定し、SQL Anywhere アプリケーションの位置を指定して実行します。SQL Anywhere インストーラは、ユーザの環境変数に対応するために *sa_config.sh* と *sa_config.csh* という 2 つのファイルを作成します。これらのファイルは、*install-dir/bin32* と *install-dir/bin64* にインストールされます。各ファイルは、必要なすべてのユーザ環境変数を設定します。

名前のとおり、1 つのファイルは Bourne シェル (*sh*) とその派生シェル (*ksh* や *bash* など) で動作するように設計されています。もう 1 つのファイルは、C シェル (*csh*) とその派生シェル (*tcsh* など) で動作するように設計されています。

一部の文は、各バッチ・ファイルでコメント・アウトされます。システム管理者は、システムの設定に応じてこれらのファイルを編集し、セクションのコメントを解除できます。

SQL Anywhere アプリケーションは、いつくかの方法で実行できます。

1. *sa_config* ファイルに指定されている環境変数をシステム環境に追加した場合は、アプリケーションを X-Window サーバなどの GUI から起動したり、ターミナル・ウィンドウでアプリケーション名を入力したりできます。
2. *sa_config* ファイルのいずれかのソースを指定してある場合は、ターミナル・ウィンドウでアプリケーション名を入力することでアプリケーションを実行できます。
3. *install-dir/bin32s* と *install-dir/bin64s* には、SQL Anywhere アプリケーションと同じ名前のスクリプトが含まれています。これらのスクリプトは、適切な環境変数を設定してから、アプリケーションを起動します。このため、対応するスクリプトを実行することで、アプリケーションを実行できます。スクリプトを実行する前に *sa_config* ファイルのソースを指定する必要はありません。

UNIX と Mac OS X でのファイルのソースの指定

ファイルの**ソースを指定する**とは、シェルの現在のインスタンスでテキスト・ファイルに含まれるコマンドを実行することを指します。これは、シェルに組み込まれたコマンドを使用することで実行できます。

Bourne シェルとその派生シェルでは、このコマンド名は . (単一のピリオド) です。たとえば、SQL Anywhere が */opt/sqlanywhere10* にインストールされている場合、次の文を使用して *sa_config.sh* のソースを指定します。

```
. /opt/sqlanywhere10/bin32/sa_config.sh
```

C シェルとその派生シェルの場合、コマンドは source です。たとえば、SQL Anywhere が */opt/sqlanywhere10* にインストールされている場合、次の文を使用して *sa_config.csh* のソースを指定します。

```
source  /opt/sqlanywhere10/bin32/sa_config.csh
```

DYLD_LIBRARY_PATH 環境変数 [Mac OS X]

Mac OS X 上で SQL Anywhere アプリケーションが必要とするライブラリの実行時の検索場所になるディレクトリを指定します。

構文

DYLD_LIBRARY_PATH=*path-list*

デフォルト

/Applications/SQLAnywhere10/System/lib32

備考

インストーラによって作成される *sa_config.sh* ファイルと *sa_config.csh* ファイルは、これらの環境変数を作成または変更するスクリプトです。

参照

- 「LD_LIBRARY_PATH 環境変数 [Linux と Solaris]」 308 ページ
- 「LIBPATH 環境変数 [AIX]」 308 ページ

- 「SHLIB_PATH 環境変数 [HP-UX]」 315 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

LD_LIBRARY_PATH 環境変数 [Linux と Solaris]

Linux と Solaris 上で SQL Anywhere アプリケーションが必要とするライブラリの実行時の検索場所になるディレクトリを指定します。

構文

LD_LIBRARY_PATH=*path-list*

デフォルト

- */opt/sqlanywhere10/lib32* (32 ビットのプラットフォーム)
- */opt/sqlanywhere10/lib64* (64 ビットのプラットフォーム)

備考

インストーラによって作成される *sa_config.sh* ファイルと *sa_config.csh* ファイルは、これらの環境変数を作成または変更するスクリプトです。

参照

- 「DYLD_LIBRARY_PATH 環境変数 [Mac OS X]」 307 ページ
- 「LIBPATH 環境変数 [AIX]」 308 ページ
- 「SHLIB_PATH 環境変数 [HP-UX]」 315 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

LIBPATH 環境変数 [AIX]

AIX 上で SQL Anywhere アプリケーションが必要とするライブラリの実行時の検索場所になるディレクトリを指定します。

構文

LIBPATH=*path-list*

デフォルト

- */usr/lpp/sqlanywhere10/lib32* (32 ビットのプラットフォーム)
- */usr/lpp/sqlanywhere10/lib64* (64 ビットのプラットフォーム)

備考

インストーラによって作成される *sa_config.sh* ファイルと *sa_config.csh* ファイルは、これらの環境変数を作成または変更するスクリプトです。

参照

- 「DYLD_LIBRARY_PATH 環境変数 [Mac OS X]」 307 ページ
- 「LD_LIBRARY_PATH 環境変数 [Linux と Solaris]」 308 ページ
- 「SHLIB_PATH 環境変数 [HP-UX]」 315 ページ

- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

ODBCHOME 環境変数 [UNIX]

.odbc.ini ファイルのロケーションを指定します。

構文

ODBCHOME=*odbc-ini-directory*

備考

.odbc.ini ファイルは、ODBC データ・ソースを含むシステム情報ファイルです。このファイルの名前が *.odbc.ini* ではない場合は、ODBCINI または ODBC_INI 環境変数を使用してロケーションを指定する必要があります。

ODBC データ・ソースの検索アルゴリズムの詳細については、「UNIX での ODBC データ・ソースの使用」 81 ページを参照してください。

参照

- 「ODBCINI と ODBC_INI 環境変数 [UNIX]」 309 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

ODBCINI と ODBC_INI 環境変数 [UNIX]

ODBC データ・ソースを含むシステム情報ファイルのパスと名前を指定します。

構文

ODBCINI=*odbc-ini-file*

ODBC_INI=*odbc-ini-file*

備考

システム情報ファイルは、これらのいずれかの環境変数を使用して指定している場合は、*.odbc.ini* という名前でなくてもかまいません。2 つの環境変数が用意されているのは、他製品との互換性を確保するためです。

ODBC データ・ソースの検索アルゴリズムの詳細については、「UNIX での ODBC データ・ソースの使用」 81 ページを参照してください。

参照

- 「ODBCHOME 環境変数 [UNIX]」 309 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

PATH 環境変数

SQL Anywhere 実行プログラムがあるディレクトリのロケーションを指定します。

構文

PATH=*path-list*

デフォルト

注意：次のパスは、対応するコンポーネントがインストールされた場合にのみ追加されます。

オペレーティング・システム	デフォルトのロケーション
Windows 32 ビット	*C:¥Program Files¥SQL Anywhere 10¥win32*
Windows ia64/Itanium	*C:¥Program Files¥SQL Anywhere 10¥ia64*
Windows x64	*C:¥Program Files¥SQL Anywhere 10¥x64*
Windows	*C:¥Program Files¥SQL Anywhere 10¥Sybase Central 5.0.0¥win32*
Mac OS X	*/Applications/SQLAnywhere10/System/bin32*
	/Applications/SQLAnywhere10/System/sybcentral500
	/Applications/SQLAnywhere10/System/openserver/OCS-15_0
AIX (32 ビット)	*/usr/lpp/sqlanywhere10/bin32*
AIX (64 ビット)	*/usr/lpp/sqlanywhere10/bin64*
AIX (すべてのプラットフォーム)	*usr/lpp/sqlanywhere10/openserver/OCS-15_0/bin*
その他の UNIX オペレーティング・システム (32 ビット)	*/opt/sqlanywhere10/bin32*
その他の UNIX オペレーティング・システム (64 ビット)	*/opt/sqlanywhere10/bin64*
その他すべての UNIX オペレーティング・システム	*/opt/sqlanywhere10/sybcentral500*
	/opt/sqlanywhere10/openserver/OCS-15_0/bin

備考

Windows では、PATH 環境変数の設定がインストーラによって変更され、SQL Anywhere 実行プログラムの保存先ディレクトリが追加されます。

UNIX のインストーラによって作成される *sa_config.sh* ファイルと *sa_config.csh* ファイルは、これらの環境変数を作成または変更するスクリプトです。

参照

- 「Windows での環境変数の設定」 306 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

SACHARSET 環境変数

SQL Anywhere が使用する文字セットを指定します。

構文

SACHARSET=*charset*

備考

charset は文字セット名です。

推奨される文字セットの詳細については、「推奨文字セットと照合」 361 ページを参照してください。

SACHARSET を指定しなかった場合は、オペレーティング・システムの文字セットが使用されます。

参照

- 「Windows での環境変数の設定」 306 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

SADIAGDIR 環境変数

SQL Anywhere 診断ディレクトリのロケーションを指定します。

構文

SADIAGDIR=*diagnostic-information-directory*

デフォルト

オペレーティング・システム	デフォルトのロケーション
Windows	*%ALLUSERSPROFILE%¥Application Data¥SQL Anywhere 10 ¥diagnostics*
UNIX	*$HOME/.sqlanywhere10/diagnostics*
NetWare	*SYS:¥SYSTEM*
Windows CE	データベース・サーバが実行されているディレクトリ

備考

SQL Anywhere では、クラッシュ・レポートと機能診断情報が診断ディレクトリに保存されます。クラッシュ・レポートが書き込まれる診断ディレクトリのロケーションは、SADIAGDIR 環境変数の設定によって決まります。

環境変数で指定されたディレクトリが存在しない場合、データベース・サーバは環境変数が設定されていないものとして処理を実行します。

Windows CE 以外の Windows では、診断情報は次の中から最初に検出された書き込み可能ディレクトリに書き込まれます。

1. SADIAGDIR 環境変数で指定されたディレクトリ
2. 現在の実行プログラムがあるディレクトリ
3. 現在のディレクトリ
4. テンポラリ・ディレクトリ「SATMP 環境変数」 314 ページと「TMP、TEMPDIR、TEMP 環境変数」 319 ページを参照してください。

Windows CE では、診断情報は次の中から最初に検出された書き込み可能ディレクトリに書き込まれます。

1. 現在の実行プログラムがあるディレクトリ。
2. 現在のディレクトリ。
3. テンポラリ・ディレクトリ。「Windows CE でのレジストリ設定」 330 ページを参照してください。

UNIX では、診断情報は次の中から最初に検出された書き込み可能ディレクトリに書き込まれます。

1. SADIAGDIR 環境変数で指定されたディレクトリ。
2. *$HOME/.sqlanywhere10/diagnostics* で指定されたディレクトリ。
3. 現在のディレクトリ。
4. テンポラリ・ディレクトリ。「SATMP 環境変数」 314 ページと「TMP、TEMPDIR、TEMP 環境変数」 319 ページを参照してください。

注意
UNIX では、データベース・サーバまたは Mobile Link サーバをデーモンとして実行する場合やユーザが root/nobody である場合、ユーザのホーム・ディレクトリにクラッシュ・レポートを書き込むことはおすすめしません。そのため、UNIX のインストール時に適切なロケーションの指定を求められ、*sa_config.sh* ファイルと *sa_config.csh* ファイルに SADIAGDIR 環境変数が設定されます。

参照

- 「SQL Anywhere サポート・ユーティリティ (dbsupport)」 722 ページ
- 「SQL Anywhere のエラー・レポート」 56 ページ
- 「Windows での環境変数の設定」 306 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

SALANG 環境変数

SQL Anywhere の言語コードを指定します。

構文

SALANG=*language-code*

備考

language-code は言語を表す 2 文字のコードです。たとえば、**SALANG=DE** の場合、デフォルトの言語がドイツ語に設定されます。

サポートされている言語コードの詳細については、「ロケール言語の知識」 346 ページを参照してください。

次の方法で最初に返された値によってデフォルトの言語が決まります。

1. SALANG 環境変数のチェック
2. (Windows の場合) レジストリが設定どおりかどうかを、インストール中、または *dblang.exe* を実行してチェックしてください。「言語選択ユーティリティ (dblang)」 678 ページを参照してください。
3. オペレーティング・システムに対する言語情報の照会
4. 言語情報を指定しない場合は、英語がデフォルトになります。

参照

- 「言語選択ユーティリティ (dblang)」 678 ページ
- 「インストール時のレジストリ設定」 329 ページ
- 「Windows での環境変数の設定」 306 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

SALOGDIR 環境変数

backup.syb ファイルのロケーションを指定します。

構文

SALOGDIR=*directory-name*

備考

SALOGDIR 環境変数が設定されている場合、バックアップ履歴ファイル *backup.syb* が格納されるディレクトリのパスが含まれることが想定されます。このファイルは BACKUP 文または RESTORE 文を実行するたびに更新されます。

Windows では、次のうち最初の書き込み可能なロケーションに *backup.syb* ファイルが作成されます。

1. SALOGDIR 環境変数

2. インストール・ディレクトリ

   32 ビットの Windows プラットフォームでは、デフォルト・ロケーションは *install-dir ¥win32* となります。このディレクトリが存在しない場合は、エラーが返されます。

3. データベース・サーバ実行プログラムのディレクトリ

4. 現在のドライブのルート・ディレクトリに *backup.syb* ファイルを書き込みます。

UNIX では、次のうち最初の書き込み可能なロケーションに *backup.syb* ファイルが作成されます。

1. SALOGDIR 環境変数

2. HOME 環境変数

3. ディレクトリ・サーバの起動ディレクトリに *backup.syb* ファイルを書き込みます。

参照

- 「BACKUP 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「Windows での環境変数の設定」 306 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

SATMP 環境変数

データベース・サーバとテンポラリ・ディレクトリを必要とする SQL Anywhere コマンド・ライン・ユーティリティが使用するテンポラリ・ファイルのロケーションを指定します。

構文

SATMP=*directory-name*

備考

SATMP 環境変数には、データベース・サーバとテンポラリ・ディレクトリを必要とする SQL Anywhere コマンド・ライン・ユーティリティが使用するテンポラリ・ファイルのロケーションが指定されます。このファイルは、データベース・サーバをサービスとして実行する場合に役立ちます。テンポラリ・ファイルをほかのプログラムがアクセスできないディレクトリに保存できるからです。

データベース・サーバの起動時に -dt オプションでテンポラリ・ファイルのロケーションを指定しなかった場合は、SATMP 環境変数の値に基づいてテンポラリ・ファイルの保存場所が決定されます。SATMP 環境変数が設定されていない場合は、環境変数 TMP、TMPDIR、TEMP のうち、最初に見つかった変数の値が使用されます。UNIX では、これらの環境変数のいずれも存在しない場合、*/tmp* が使用されます。

UNIX では、共有メモリを介して接続する場合、クライアントとデータベース・サーバの両方で SATMP を同じ値に設定する必要があります。

UNIX の共有メモリ接続の保護の詳細については、「セキュリティのヒント」 920 ページを参照してください。

Windows CE では、サーバのテンポラリ・ディレクトリとして使用するディレクトリをレジストリに指定できます。

Windows CE 上のテンポラリ・ファイルのロケーションについては、「Windows CE でのレジストリ設定」 330 ページを参照してください。

参照

- 「-dt サーバ・オプション」 160 ページ
- 「TMP、TEMPDIR、TEMP 環境変数」 319 ページ
- 「Windows での環境変数の設定」 306 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ
- 「異なるファイルの異なるデバイスへの配置」 『SQL Anywhere サーバ - SQL の使用法』

旧バージョンのソフトウェアでの共有メモリ接続の使用
SQL Anywhere バージョン 9 以前では、環境変数 ASTMP が SATMP に相当します。共有メモリを使用してバージョン 9 とバージョン 10 のソフトウェアに接続する場合は、SATMP と ASTMP 環境変数を設定して、テンポラリ・ディレクトリとして (同じ) ロケーションを指定する必要があります。

SHLIB_PATH 環境変数 [HP-UX]

HP-UX 上で SQL Anywhere アプリケーションが必要とするライブラリの実行時の検索場所になるディレクトリを指定します。

構文

SHLIB_PATH=*path-list*

デフォルト

- */opt/sqlanywhere10/lib32* (32 ビットのプラットフォーム)
- */opt/sqlanywhere10/lib64* (64 ビットのプラットフォーム)

備考

インストーラによって作成される *sa_config.sh* ファイルと *sa_config.csh* ファイルは、これらの環境変数を作成または変更するスクリプトです。

参照

- 「DYLD_LIBRARY_PATH 環境変数 [Mac OS X]」 307 ページ
- 「LD_LIBRARY_PATH 環境変数 [Linux と Solaris]」 308 ページ
- 「LIBPATH 環境変数 [AIX]」 308 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

SQLANY10 環境変数

SQL Anywhere 10 を含むディレクトリののロケーションを指定します。

構文

SQLANY10=*directory-name*

デフォルト

オペレーティング・システム	ロケーション
Windows	*C:¥Program Files¥SQL Anywhere 10*
AIX	*/usr/lpp/sqlanywhere10*
Mac OS X	*/Applications/SQLAnywhere10/System*
その他の UNIX オペレーティング・システム	*/opt/sqlanywhere10*

備考

この環境変数は必ず設定してください。これには、いくつかの理由があります。その 1 つとして、サンプルが SQL Anywhere アプリケーションを探すときに、この環境変数が必要となることがあります。

Windows では、SQLANY10 環境変数のロケーションがインストーラによって設定されます。

UNIX のインストーラによって作成される *sa_config.sh* ファイルと *sa_config.csh* ファイルは、これらの環境変数を作成または変更するスクリプトです。

参照

- 「Windows での環境変数の設定」 306 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

SQLANYSAMP10 環境変数

SQL Anywhere サンプル・ディレクトリのロケーションを指定します。

構文

SQLANYSAMP10=*directory-name*

デフォルト

オペレーティング・システム	デフォルトのロケーション
Windows	*C:¥Documents and Settings¥All Users¥Documents ¥SQL Anywhere 10¥Samples*
Windows Vista	*C:¥Users¥Public¥Documents¥SQL Anywhere 10 ¥Samples*
Mac OS X	*/Applications/SQLAnywhere10/Samples*
AIX	*/usr/lpp/sqlanywhere10/samples*

オペレーティング・システム	デフォルトのロケーション
その他の UNIX オペレーティング・システム	*/opt/sqlanywhere10/samples*

備考

Windows では、SQLANYSAMP10 環境変数のロケーションがインストーラによって設定されます。

UNIX のインストーラによって作成される *sa_config.sh* ファイルと *sa_config.csh* ファイルは、これらの環境変数を作成または変更するスクリプトです。

参照

- 「Windows での環境変数の設定」 306 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

SQLANY10 環境変数

共有コンポーネントのディレクトリのロケーションを指定します。

構文

SQLANYSH10=*directory-name*

デフォルト

オペレーティング・システム	デフォルトのロケーション
Windows	*C:¥Program Files¥SQL Anywhere 10*
Mac OS X	*/Applications/SQLAnywhere10/System*
AIX	*/usr/lpp/sqlanywhere10*
その他の UNIX オペレーティング・システム	*/opt/sqlanywhere10*

備考

Interactive SQL、Sybase Central、SQL Anywhere コンソール・ユーティリティ (dbconsole) は、この変数の値に基づいて共有コンポーネントのディレクトリのロケーションを決定します。

Windows では、SQLANYSH10 環境変数のロケーションがインストーラによって設定されます。

UNIX のインストーラによって作成される *sa_config.sh* ファイルと *sa_config.csh* ファイルは、これらの環境変数を作成または変更するスクリプトです。

参照

- 「Windows での環境変数の設定」 306 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

SQLCONNECT 環境変数

データベース・サーバに接続するときにデータベース管理ユーティリティが使用する接続パラメータを指定します。

構文

SQLCONNECT=*parameter*=*value*; ⋯

備考

これは、**parameter**=*value* の形式のパラメータ設定をセミコロンで区切ったリストの文字列で指定します。

サポートされている接続パラメータの詳細については、「接続パラメータ」 230 ページを参照してください。

パスワードのセキュリティ・リスク
パスワードはプレーン・テキストなので、それを SQLCONNECT 環境変数に含めることはセキュリティ・リスクとなります。

参照

- 「Windows での環境変数の設定」 306 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

SQLPATH 環境変数

コマンド・ファイルとヘルプ・ファイルのロケーションを指定します。

構文

SQLPATH=*path-list*

備考

Interactive SQL は、システム・パスを検索する前に、SQLPATH に指定されているディレクトリでコマンド・ファイルとヘルプ・ファイルを検索します。

参照

- 「Windows での環境変数の設定」 306 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

SQLREMOTE 環境変数

SQL Remote 中の FILE メッセージ・リンクのアドレスである、サブディレクトリを指定します。

構文

SQLREMOTE=*path*

備考

SQL Remote 中の FILE メッセージ・リンクのアドレスは、SQLREMOTE 環境変数のサブディレクトリです。この環境変数には共有ディレクトリを指定してください。

Windows オペレーティング・システム (Windows CE を除く) では、SQLREMOTE 環境変数を設定する代わりに、*SQL Remote¥Directory* レジストリ・エントリを適切なルート・ディレクトリに設定することもできます。

参照

- 「Windows での環境変数の設定」 306 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

SYBASE 環境変数

Adaptive Server Enterprise、Open Client、Open Server などの Sybase アプリケーションや DSEdit などのユーティリティのインストール用ホーム・ディレクトリを指定します。

構文

SYBASE=*directory-name*

備考

この環境変数が必要になるのは、その他の Sybase アプリケーションを使用する場合のみです。

参照

- 「Windows での環境変数の設定」 306 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ

TMP、TEMPDIR、TEMP 環境変数

SQL Anywhere テンポラリ・ファイルのロケーションを指定します。

構文

TMP=*path*

TMPDIR=*path*

TEMP=*path*

備考

SQL Anywhere ソフトウェアにより、各種操作に使用するテンポラリ・ファイルが作成される場合があります。テンポラリ・ファイルは、環境変数 TMP、TMPDIR、TEMP のいずれかで指定されたディレクトリに作成されます。これらの環境変数が重複指定されている場合は、TMP、TMPDIR、TEMP の中で最初に出現したものが採用されます。

SQL Anywhere サーバは、SATMP 環境変数を最初にチェックします。これが指定されていなかった場合に、上記の環境変数がチェックされます。「SATMP 環境変数」 314 ページを参照してください。

これらの環境変数が定義されていない場合は、テンポラリ・ファイルは、サーバが現在作業中のディレクトリに配置されます。UNIX では、これらの環境変数のいずれも見つからない場合は、*/tmp* が使用されます。

Windows CE では、サーバのテンポラリ・ディレクトリとして使用するディレクトリをレジストリに指定できます。

テンポラリ・ディレクトリ値の設定については、「Windows CE でのレジストリ設定」 330 ページを参照してください。

旧バージョンのソフトウェアでの共有メモリ接続の使用
SQL Anywhere バージョン 9 以前では、環境変数 ASTMP が SATMP に相当します。共有メモリを使用してバージョン 9 とバージョン 10 のソフトウェアに接続する場合は、SATMP と ASTMP 環境変数を設定して、テンポラリ・ファイルのロケーションを指定する必要があります。

参照

- 「-dt サーバ・オプション」 160 ページ
- 「SATMP 環境変数」 314 ページ
- 「Windows での環境変数の設定」 306 ページ
- 「UNIX と Mac OS X での環境変数の設定」 306 ページ
- 「異なるファイルの異なるデバイスへの配置」 『SQL Anywhere サーバ - SQL の使用法』

第 9 章

ファイル・ロケーションとインストール設定

目次

インストール・ディレクトリ構造

SQL Anywhere をインストールすると、いくつかのディレクトリが作成されます。このディレクトリ内のファイルの中には、不可欠なものもあり、そうでないものもあります。この項ではディレクトリ構造について説明します。

1 つの製品として入手した場合も、他の製品の一部として入手した場合も、SQL Anywhere ソフトウェアは単一のインストール・ディレクトリにインストールされます。SQL Anywhere の付属ユーティリティは、他のディレクトリにインストールされます。この項では、SQL Anywhere 自体のインストール・ディレクトリ構造について説明します。SQLANY10 環境変数は、インストール・ディレクトリのロケーションを指定します。「SQLANY10 環境変数」 315 ページを参照してください。

SQL Anywhere インストール・ディレクトリ

SQL Anywhere のインストール・ディレクトリには、次のようなものがあります。

- **『はじめにお読みください』** *readme.txt* ファイルには、最新の情報が記載されています。

NetWare と Windows CE 以外のプラットフォームでは、インストール・ディレクトリの下にいくつかのディレクトリがあります。

- **実行ディレクトリ** 各オペレーティング・システム・プラットフォームごとに別々のディレクトリがあり、それぞれに実行プログラム、ダイナミック・リンク・ライブラリ、ヘルプ・ファイルが保存されています。

  Windows CE 以外の Windows の場合、これらのファイルは *win32*、*ia64*、または *x64* ディレクトリにインストールされます。UNIX を使用している場合は、これらのファイルは *bin32* または *bin64* ディレクトリと *lib32* または *lib64* ディレクトリにインストールされます。.

  使用するコンピュータのオペレーティング・システムのバージョンに合ったディレクトリだけが作成されます。

- **java ディレクトリ**
  このディレクトリには Java クラスが保存されます。

- **scripts ディレクトリ**
  スクリプト・ディレクトリには、データベース管理ユーティリティによって使用され、サンプルとしても使用される SQL スクリプトがあります。スクリプト・ディレクトリがないと、管理ユーティリティは動作しません。

- **h ディレクトリ** *h* ディレクトリには、SQL Anywhere 対応の C/C++ アプリケーションを開発するためのヘッダ・ファイルが保存されます。UNIX では、このディレクトリは *include* と呼ばれます。

Novell NetWare ファイルのロケーション

Novell NetWare では、すべてのファイルがサーバ上の単一ディレクトリにインストールされます。このマニュアルで、インストール・ディレクトリのサブディレクトリにあるファイルについて説明している場合、NetWare では、そのファイルはインストール・ディレクトリにあります。

Windows CE ファイルのロケーション

Windows CE では、すべてのファイルがインストール・ディレクトリ *¥Program Files¥SQLAny10* に保存されます。ただし、DLL は *¥Windows* ディレクトリに保存されます。サブディレクトリは作成されません。

UNIX ファイルのロケーション

言語リソースは *res* ディレクトリにインストールされ、共有オブジェクトは *lib32* または *lib64* ディレクトリにインストールされます。

サンプル・ディレクトリ

SQL Anywhere 10 をインストールするとき、サンプルのインストール先ディレクトリを選択できます。このマニュアルではそのロケーションを *samples-dir* と表します。

SQLANYSAMP10 環境変数は、*samples-dir* のロケーションを指定します。「SQLANYSAMP10 環境変数」 316 ページを参照してください。

Windows では、[スタート] メニューから [プログラム] – [SQL Anywhere 10] – [サンプル・アプリケーションおよびプロジェクト] を選択することでサンプルにアクセスできます。

次の表に、サポートされている各オペレーティング・システムの *samples-dir* のデフォルト・ロケーションと一般的なロケーションを示します。

オペレーティング・システム	デフォルトのインストール先 (*samples-dir*)	一般的なインストール先
Windows XP/200x	*%ALLUSERSPROFILE%¥Documents ¥SQL Anywhere 10¥Samples*	*C:¥Documents and Settings¥All Users ¥Documents¥SQL Anywhere 10 ¥Samples*[1]
Windows Vista	*%PUBLIC%¥Documents¥SQL Anywhere 10¥Samples*	*C:¥Users¥Public¥Documents¥SQL Anywhere 10¥Samples*
Windows CE	*¥Program Files¥SQLAny10*	
UNIX と Linux	*/opt/sqlanywhere10/samples*	サンプルをユーザ指定ディレクトリにコピーするスクリプトが用意されています。
Mac OS X	*/Applications/SQLAnywhere10/Samples*	サンプルをユーザ指定ディレクトリにコピーするスクリプトが用意されています。
NetWare	サンプルのインストールなし	

[1] SQL Anywhere サンプル・ディレクトリに Windows エクスプローラでアクセスする場合、ロケーションは *Documents and Settings* > *All Users* > ***Shared Documents*** > *SQL Anywhere 10* > *Samples* です。ただし、SQL Anywhere サンプル・ディレクトリにコマンド・プロンプトからアクセスする場合のパスは、*C:¥Documents and Settings¥All Users¥Documents¥SQL Anywhere 10¥Samples* です。

SQL Anywhere のファイル検索方法

クライアント・ライブラリとデータベース・サーバは、主に次の 2 つの理由からファイルを検索する必要があります。

- SQL Anywhere を実行するためには、DLL と初期化ファイルが必要です。不正な DLL が検索されると、バージョン・ミスマッチ・エラーが発生する可能性があります。
- INSTALL JAVA や LOAD TABLE など、SQL 文で指定して、ランタイムに検索する必要があるものがあります。

ファイル名を使用する SQL 文の例は次のとおりです。

- **INSTALL JAVA 文** Java クラスを保存するファイル名です。
- **LOAD TABLE 文と UNLOAD TABLE 文** データがロードまたはアンロードされるファイル名です。
- **CREATE DATABASE 文** この文と、ファイルを作成できる同様の文には、ファイル名が必要です。

SQL Anywhere は簡易アルゴリズムを使用してファイルを検索する場合があります。それ以外の場合、より広範囲の検索が行われます。

簡単なファイル検索

多くの SQL 文 (LOAD TABLE や CREATE DATABASE など) では、ファイル名はデータベース・サーバの現在作業中のディレクトリに対する相対名として解釈されます。

また、データベース・サーバが起動され、データベース・ファイル名 (DatabaseFile (DBF) パラメータ) が提供されると、パスは現在作業中のディレクトリに対する相対名として解釈されます。

Windows での広範囲なファイル検索

Windows CE を除く Windows では、データベース・サーバと管理ユーティリティを含む SQL Anywhere プログラムは、DLL や共有ライブラリなど、必要なファイルをより広範囲に検索します。この場合、SQL Anywhere プログラムは次の順番でファイルを検索します。

1. 実行プログラム・ディレクトリ (プログラムの実行ファイルがあるディレクトリ)
2. 関連ディレクトリ。実行プログラム・ディレクトリに対する次の相対パスを持つディレクトリのことです。
   - 実行ディレクトリの親
   - *scripts* という親ディレクトリの子。
3. 現在の作業ディレクトリ起動したプログラムは、現在作業中のディレクトリを持つことになります (プログラムの起動元のディレクトリ)。必要なファイルは、このディレクトリ内で検索されます。

4. ロケーション・レジストリ・エントリ。Windows へのインストール時に、SQL Anywhere は Location レジストリ・エントリを追加します。指定されたディレクトリに続いて、次のディレクトリが検索されます。
   - *scripts* という名前のサブディレクトリ。
   - オペレーティング・システム名の付いたサブディレクトリ (*win32*、*x64* など)
5. システムに応じたディレクトリ。このディレクトリには、一般的なオペレーティング・システム・ファイルが格納されているディレクトリが含まれます。たとえば Windows では、*Windows* ディレクトリや *Windows¥system32* ディレクトリになります。
6. CLASSPATH ディレクトリ。Java ファイルの場合は、CLASSPATH 環境変数で指定されているディレクトリでファイルを検索します。
7. PATH ディレクトリ。システム・パスとユーザ・パスのディレクトリでファイルを検索します。

Windows CE での広範囲なファイル検索

Windows CE では、データベース・サーバと管理ユーティリティを含む SQL Anywhere プログラムは、DLL などの必要なファイルをより広範囲に検索します。この場合、SQL Anywhere プログラムは次の順番でファイルを検索します。

1. 実行プログラム・ディレクトリ (プログラムの実行ファイルがあるディレクトリ)
2. ¥ (ルート・ディレクトリ)
3. Location レジストリ・エントリ (レジストリ・キー *HKEY_LOCAL_MACHINE¥SOFTWARE¥Sybase¥SQL Anywhere¥10.0¥Location* で指定されたディレクトリ)
4. *¥Windows* ディレクトリ

UNIX での広範囲なファイル検索

UNIX では、データベース・サーバと管理ユーティリティを含む SQL Anywhere プログラムは、DLL や共有ライブラリなど、必要なファイルをより広範囲に検索します。この場合、SQL Anywhere プログラムは次の順番でファイルを検索します。

1. サーバの現在の作業ディレクトリ (*./*) からの相対パス
   a. *./res*
   b. *./scripts*
   c. *./tix*
   d. *./bin*
   e. *./lib*
   f. *./java*
   g. *./shared*

h. *./../shared*

> **注意**
> これらのサブディレクトリは、手順 6 を除くすべての手順で検索されます。

2. 実行パス
3. SYBASE 環境変数
4. PATH 環境変数
5. LIBPATH 環境変数

   - LD_LIBRARY_PATH (Linux と Solaris)
   - SHLIB_PATH (HP-UX)
   - LIBPATH (AIX)
   - DYLD_LIBRARY_PATH (Mac OS X)

6. 以下のいずれか

   a. *install-dir/scripts* (*.sql* タイプのファイル)

   b. *install-dir/java* (*.zip* タイプのファイル)

   c. *install-dir/charsets/unicode* (*.uct* タイプのファイル)

7. SQL Anywhere インストール・ディレクトリ (*install-dir*)。*install-dir* は SQLANY*x* 環境変数 (定義されている場合) で指定された単一のディレクトリです。

NetWare での広範囲なファイル検索

NetWare では、データベース・サーバと管理ユーティリティを含む SQL Anywhere プログラムは、必要なファイルをより広範囲に検索します。この場合、SQL Anywhere プログラムは次の順番でファイルを検索します。

1. 実行プログラム・ディレクトリ (プログラムの実行ファイルがあるディレクトリ)
2. 現在の作業ディレクトリ
3. 実行プログラム・ディレクトリの兄弟ディレクトリ *scripts* (*.sql* ファイルの場合)
4. 実行プログラム・ディレクトリの兄弟ディレクトリ *java* (*.zip* ファイルまたは *.jar* ファイルの場合)
5. 実行プログラム・ディレクトリの親ディレクトリ
6. *SYS:SYSTEM* ディレクトリ
7. *SYS:SYSTEM¥scripts* ディレクトリ (*.sql* ファイルの場合)
8. *SYS:SYSTEM¥java* ディレクトリ (*.zip* ファイルまたは *.jar* ファイルの場合)

レジストリと INI ファイル

Windows オペレーティング・システム (Windows CE 以外) では、SQL Anywhere はレジストリ設定を使用します。UNIX と NetWare の場合、この設定は代わりに初期化ファイルで行います。

これらの設定はソフトウェアのインストール中に行われるので、通常はユーザがレジストリや初期化ファイルにアクセスする必要はありません。ここでは、オペレーティング環境を変更するユーザのために説明します。

SQL Anywhere で使用する *.ini* ファイルの内容は、ファイル非表示ユーティリティを使用した単純暗号化によって難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。

警告
SQL Anywhere データ・ソースだけを使用している場合を除き、UNIX 上でファイル非表示ユーティリティ (dbfhide) を使用して、システム情報ファイル (デフォルトのファイル名は *.odbc.ini*) に単純暗号化を追加しないでください。他のデータ・ソース (Mobile Link 同期など) を使用する予定の場合、システム情報ファイルの内容を難読化すると、他のドライバが正しく機能しなくなることがあります。

現在のユーザとローカル・マシン設定

オペレーティング・システムによっては、2 つのレベルのシステム設定が存在することがあります。一部の設定は個々のユーザに固有であり、そのユーザがログオンしたときだけ使用できます。これらの設定を「**現在のユーザ**」設定と呼びます。また、コンピュータ全体に関連し、どのユーザがログオンしているかに関わらず使用できるものを、「**ローカル・マシン**」設定と呼びます。ローカル・マシン設定を変更するには、コンピュータの管理者パーミッションを取得する必要があります。

SQL Anywhere は、現在のユーザ設定とローカル・マシン設定の両方を許可します。たとえば、Windows XP では、それぞれ *HKEY_CURRENT_USER* キーと *HKEY_LOCAL_MACHINE* キーに保存されます。

現在のユーザを優先

現在のユーザとローカル・マシン・レジストリの両方に設定がある場合は、現在のユーザ設定がローカル・マシン設定より優先されます。

ローカル・マシン設定が必要なとき

SQL Anywhere プログラムを「**サービス**」として実行する場合は、設定が「*ローカル・マシン*」レベルで行われていることを確認してください。

サービスは、コンピュータ全体を停止しないかぎり、コンピュータからログオフしても特別なアカウントで実行を継続できます。サービスは、個々のアカウントに依存しないようにできます。そのため、ローカル・マシン設定にアクセスすることが必要です。

SQL Anywhere プログラムの他に、一部の Web サーバがサービスとして実行されます。そのような Web サーバで Apache や IIS を使用するには、ローカル・マシン設定をしてください。

一般的には、ローカル・マシン設定をおすすめします。

レジストリ構造

Windows (Windows CE 以外) では、レジストリ・エディタでレジストリに直接アクセスできます。SQL Anywhere レジストリ・エントリは、以下のロケーションにある *HKEY_CURRENT_USER* または *HKEY_LOCAL_MACHINE* キーに保管されています。

```
Software
  Sybase
    SQL Anywhere
      10.0
    Sybase Central
      5.0.0
```

レジストリの変更の危険性
レジストリは、ユーザ自身の責任で変更してください。レジストリを変更する前に、システムのバックアップをとることをおすすめします。

インストール時のレジストリ設定

Windows では、*HKEY_LOCAL_MACHINE¥Software¥Sybase* レジストリ内の次のレジストリ設定は、インストール・プログラムによって行われます。

- **SQL Anywhere¥10.0¥Location**　このエントリには、SQL Anywhere ソフトウェアのインストール・ディレクトリが設定されます。次に例を示します。

  ```
  Location "c:¥Program Files¥SQL Anywhere 10"
  ```

- **SQL Anywhere¥10.0¥Shared Location**　このエントリには、Sybase Central や Sun Java Runtime Environment (JRE) などの共有ソフトウェアのインストール・ディレクトリが設定されます。次に例を示します。

  ```
  Location "c:¥Program Files¥SQL Anywhere 10"
  ```

  このエントリは Interactive SQL によって使用されます。

- **SQL Anywhere¥10.0¥Folder**　このエントリは [スタート メニュー] フォルダの名前を示し、SQL Anywhere インストーラによって使用されます。次に例を示します。

  ```
  Folder "SQL Anywhere 10"
  ```

- **SQL Anywhere¥10.0¥Online Resources**　このエントリにはオンライン・リソースのロケーションが設定されます。次に例を示します。

  ```
  Online Resources "c:¥Program Files¥SQL Anywhere 10¥support¥ianywhere.html"
  ```


- **SQL Anywhere¥10.0¥Language**　このエントリには、メッセージとエラーに使用される現在の言語を示す 2 文字のコードが設定されます。次に例を示します。

  ```
  Language "EN"
  ```

  言語は、インストール中に指定された:言語選択に基づいて設定されます。「ロケール言語の知識」 346 ページを参照してください。

- **SQL Anywhere¥10.0¥SNMPDLL¥IniFile**　このエントリには、SNMP Extension Agent 初期化ファイルのロケーションと名前が設定されます。次に例を示します。

  ```
  IniFile "c:¥Program Files¥SQL Anywhere 10¥win32¥sasnmp.ini"
  ```

- **SQL Anywhere¥10.0¥SNMPDLL¥Pathname**　このエントリには、SNMP Extension Agent DLL のロケーションと名前が設定されます。次に例を示します。

  ```
  Pathname "c:¥Program Files¥SQL Anywhere 10¥win32¥dbsnmp10.dll"
  ```

- **Sybase Central¥5.0.0¥Location**　このエントリには、Sybase Central のインストール・ディレクトリが設定されます。次に例を示します。

  ```
  Location "c:¥Program Files¥SQL Anywhere 10¥Sybase Central 5.0.0"
  ```

  このエントリは Sybase Central によって使用されます。

- **Sybase Central¥5.0.0¥Shared Location**　このエントリには、Sybase Central や Sun Java Runtime Environment (JRE) などの共有ソフトウェアのインストール・ディレクトリが設定されます。次に例を示します。

  ```
  Location "c:¥Program Files¥SQL Anywhere 10"
  ```

  このエントリは Sybase Central によって使用されます。

- **Sybase Central¥5.0.0¥Language**　このエントリには、メッセージとエラーに使用される現在の言語を示す 2 文字のコードが設定されます。次に例を示します。

  ```
  Language "EN"
  ```

  このエントリは Sybase Central によって使用されます。言語は、インストール中に指定された:言語選択に基づいて設定されます。「ロケール言語の知識」 346 ページを参照してください。

Windows CE でのレジストリ設定

Windows CE でサーバのテンポラリ・ディレクトリとして使用するディレクトリは、次のレジストリ値で指定できます。

HKEY_CURRENT_USER¥Software¥Sybase¥SQL Anywhere¥10.0¥TempFolder

TempFolder には、使用するテンポラリ・ディレクトリの名前を指定します。サーバは次のいずれかを実行します。

- 指定のディレクトリが存在する場合、そのディレクトリを使用する。
- 指定のディレクトリがなく、親ディレクトリが存在する場合、指定のディレクトリを作成する。

指定のディレクトリがなく、作成もできない場合、データベース・サーバでは次のように処理されます。

- *¥Temp* ディレクトリが存在する場合、そのディレクトリを使用する。
- *¥Temp* ディレクトリが存在しない場合、そのディレクトリを作成する。

¥Temp ディレクトリが存在せず、作成もできない場合、サーバは現在のディレクトリを使用します。

第 10 章

国際言語と文字セット

目次

SQL Anywhere のローカライズ版

ローカライズとは、製品を目的のロケールの言語や文化に適用させることです。ロケールとは、通常は言語と国／地域の組み合わせです。ローカライズは、包装、インストーラ、マニュアル、ソフトウェアのユーザ・インタフェース、エラー／警告／情報メッセージなど、多くのコンポーネントに反映されます。

SQL Anywhere は、次の 5 言語にローカライズされており、購入いただくことができます。

- 英語
- フランス語
- ドイツ語
- 日本語
- 中国語 (簡体文字)

使用言語はインストール時に選択します。ソフトウェアとオンライン・マニュアルはこのとき指定された言語でインストールおよび設定されます。

Windows の場合、[スタート] メニューを使用して、インストールされた言語と英語を切り替えることができます。言語選択ユーティリティ (dblang) を使用すると、追加で導入した言語を含む使用可能なすべての言語に設定し直すことができます。「Windows での配備ソフトウェアのローカライズ」 335 ページと「言語選択ユーティリティ (dblang)」 678 ページを参照してください。

次の表では、各言語のサポート状況をオペレーティング・システム・プラットフォーム別に示します。

プラットフォーム	英語	ドイツ語	フランス語	日本語	中国語 (簡体文字)
Windows	○	○	○	○	○
Windows CE	○	○	○	○	○
Linux	○	○		○	○
UNIX	○				
Mac OS X	○				
NetWare	○				

ソフトウェアとマニュアルの完全ローカライズ

SQL Anywhere の Windows 版には 5 言語が用意されており、開発、配備、管理に適しています。サポート言語は次のとおりです。

- 英語
- フランス語
- ドイツ語

- 日本語
- 中国語 (簡体文字)

これらの言語では、次に示すものを含め、すべての SQL Anywhere コンポーネントがローカライズされています。

- 包装
- インストーラ
- マニュアル (HTML ヘルプ、PDF など)
- ソフトウェア
  - [スタート] メニュー項目とプログラム・フォルダ
  - データベース・サーバとクライアント・ライブラリ
  - Mobile Link サーバおよびクライアント
  - SQL Remote クライアント
  - 管理ツール (Interactive SQL、Sybase Central、およびすべての関連プラグイン)
  - コマンド・ライン・ツール (dbinit、dbunload など)

次のコンポーネントはローカライズされません。英語版のみで使用できます。

- DataWindow .NET
- InfoMaker
- PowerDesigner Physical Data Model

Windows での配備ソフトウェアのローカライズ

上記 5 言語の他に、SQL Anywhere には次の言語の配備ソフトウェア・リソースが用意されています。

- イタリア語
- 韓国語
- リトアニア語
- ポーランド語
- ブラジル・ポルトガル語
- ロシア語
- スペイン語
- 中国語 (繁体文字)
- ウクライナ語

配備ローカライズは、エンド・ユーザに配備されることが多い一部のソフトウェア・コンポーネントが対象となります。包装、マニュアル、管理ソフトウェア、開発ソフトウェア、およびイン

ストール・ソフトウェアはローカライズされません。ローカライズされるソフトウェア・コンポーネントは次のとおりです。

- データベース・サーバとクライアント・ライブラリ
- Mobile Link サーバおよびクライアント
- SQL Remote クライアント
- コマンド・ライン・ツール (dbinit、dbunload など)

SQL Anywhere の国際化機能

国際化とは、ソフトウェアの指定言語やソフトウェアを実行しているオペレーティング・システムに関係なく、ソフトウェアが各種言語とそれに適した文字セットに対処できるようにすることです。SQL Anywhere は完全に国際化に対応しています。次に、要求されたり使用されたりすることが多い機能について説明します。

- **ユニコードのサポート**　SQL Anywhere のユニコード・サポートは次のとおりです。
  - クライアントでは、ODBC、OLE DB、ADO.NET、JDBC の SQL Anywhere クライアント・ライブラリの UTF-16 をサポートしています。
  - UTF-8 のユニコード文字データの格納に NCHAR データ型を使用できます。
  - CHAR データ型では UTF-8 エンコードを使用できます。
- **コード・ページと文字セット**　SQL Anywhere データベース・サーバと関連ツールは、Windows (ANSI/ISO)、UTF-8、UNIX のコード・ページと文字セットをサポートしています。
- **照合**　SQL Anywhere では、照合アルゴリズムとして、SQL Anywhere 照合アルゴリズム (SACA) と、International Components for Unicode (ICU) を使用したユニコード照合アルゴリズム (UCA) の 2 つをサポートしています。

  ICU の詳細については、「ICU とは何か、いつ必要になるか」 337 ページを参照してください。

  SACA を使用すると、ソートが高速、簡潔、実用的になりますが、言語的な正確さが低下します。UCA を使用すると、言語的な処理は正確になりますが、記憶領域の要件と実行時間が多少増加します。「照合の知識」 349 ページを参照してください。

  高度なソートおよび比較機能として、SQL Anywhere には SORTKEY および COMPARE 関数が用意されています。これらの関数は、辞書や電話帳並みに言語的に高度なソート機能を実現します。必要に応じて、大文字小文字とアクセント記号を区別しないソートや比較ができます。「SORTKEY 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』と「COMPARE 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  SQL Anywhere には、文字カラムで SORTKEY ベースのソートを自動的に使用する設計機能も用意されています。sort_collation データベース・オプションを使用すると、文字カラムに ORDER BY が指定されたときに使用されるソート順序を指定できます。文字カラムのソート・キーの格納に計算カラムも使用できるので、ORDER BY が指定されるたびに計算カラム

を計算する必要はありません。「sort_collation オプション [データベース]」 495 ページを参照してください。

- **文字セット変換**　SQL Anywhere では、サーバ・システムとクライアント・システムとの間でデータの文字セット・エンコードを変換できます。これによって、複数の異なる文字セットを使用している環境でもデータの整合性を維持できます。「文字セット変換」 344 ページを参照してください。
- **識別子**　SQL Anywhere では、ほとんどのシングルバイト文字やマルチバイト文字を含む識別子を引用符で囲まなくても使用できます。例外は、スペースや句読表記記号です。
- **通貨**　通貨記号は、ユーロ記号も含めて、ソートの対象になります。SQL Anywhere では、通貨の書式サポートはありません。
- **日付と時刻のフォーマット**　SQL Anywhere では太陽暦を採用し、日付と時刻の設定用にさまざまなフォーマットを用意しています。カスタム・フォーマットは、date_format、time_format、timestamp_format の各データベース・オプションを使用して実現できます。date_format と timestamp_format オプションのデフォルトは、ISO 互換の日付形式 YYYY-MM-DD です。SQL Anywhere には CONVERT 関数が用意されており、日付と時刻の出力フォーマットを一般的な各種フォーマットに変換できます。

参照

- 「date_format オプション [互換性]」 441 ページ
- 「time_format オプション [互換性]」 504 ページ
- 「timestamp_format オプション [互換性]」 505 ページ
- 「CONVERT 関数 [データ型変換]」『SQL Anywhere サーバ - SQL リファレンス』

ICU とは何か、いつ必要になるか

ICU (International Components for Unicode) は、IBM が開発および保守しているオープン・ソース・ライブラリです。ICU は、ユニコード・サポートを提供することによって、ソフトウェアの国際化を容易にします。SQL Anywhere は、ICU を使用して、特定の文字セット変換と照合オペレーションを実装しています。

データベース・サーバで ICU が必要になるとき (Windows CE を除くすべてのプラットフォーム)

データベース・サーバで常に ICU を利用できることが理想的です。次の表に、ICU が必要になるときとその理由を示します。

ICU が必要になるとき	説明
1. NCHAR 文字セットまたは CHAR 文字セットの照合として UCA が使用されている	UCA は ICU を必要とします。
2. データベースの文字セットは UTF-8 でなく、マルチバイト文字セットである	データベースの文字セットから UTF-8 にパスワード変換するために必要です (データベース・パスワードは内部的に UTF-8 で格納されています)。

ICU が必要になるとき	説明
3. クライアントとデータベースの文字セットが異なり、いずれかがマルチバイト (UTF-8 を含む) である。これには、ユニコード ODBC、OLE DB、ADO.NET、および iAnywhere JDBC アプリケーションが含まれます。ICU のインストールされていないクライアントが使用するデータベース文字セットには関係ありません。	マルチバイト文字セットの適切な変換を行うためには ICU が必要です。
4. データベースの文字セットが UTF-8 でなく、CHAR と NCHAR 間の変換が必要である	データベース・サーバは、UTF-8 を別の文字セットに変換するために ICU を必要とします。
5. Embedded SQL クライアントが UTF-8 以外の NCHAR 文字セットを使用する	データベース・サーバは、UTF-8 を別の文字セットに変換するために ICU を必要とします。Embedded SQL クライアントのデフォルトの NCHAR 文字セットは、最初のクライアントの CHAR 文字セットと同じです。これは、db_change_nchar_charset 関数を使用して変更できます。「db_change_nchar_charset 関数」『SQL Anywhere サーバ - プログラミング』を参照してください。
6. CSCONVERT または SORTKEY 関数が使用されている。CSCONVERT 関数は、上記の 3 番目の項目要件に準ずる文字セットを変換するために呼び出されます。	上記の 3 番目の項目の場合、文字セットの変換には ICU が必要です。SORTKEY のラベル用に SORTKEY を生成するには UCA が必要であるため、結果として ICU が必要になります。「CSCONVERT 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』と「SORTKEY 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

データベース・サーバで ICU が必要になるとき (Windows CE)

次の表に、Windows CE で ICU が必要になるときとその理由を示します。

ICU が必要になるとき	説明
NCHAR 照合または CHAR 照合として UCA が使用されている	UCA は ICU を必要とします。
SORTKEY 関数が使用されている	SORTKEY のラベル用に SORTKEY を生成するには UCA が必要であるため、結果として ICU が必要になります。「SORTKEY 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
CHAR 文字セットが OS 文字セットに一致しない	文字セットが一致する場合であっても、NCHAR を使用しているとき、または CHAR 文字セットがマルチバイトであるときは、文字セット変換の処理を向上させるため、ICU の使用をおすすめします。

注意
ICU ライブラリをインストールしていない場合は、Windows CE の文字セットと一致する文字セットを使う照合を選択するか、データベースを作成するときに CHAR 照合として UTF8BIN 照合を選択することが必要です。また、データベースを作成するときに NCHAR 照合として UTF8BIN を選択することも必要になります。

データベース・サーバ上で ICU を使用しないで正しい文字セット変換を実行できるとき

データベースの文字セットとクライアントの文字セットの両方がシングルバイトであり、*sqlany.cvf* を利用できるとき (すべてのプラットフォーム)、またはオペレーティング・システムが変換をサポートするとき (Windows のみ) は、ICU を使用しないで正しい文字セット変換を行うことができます。これは、*sqlany.cvf* を利用できる場合はシングルバイト間の変換は ICU で処理可能であるからです。また、ホストのオペレーティング・システムには適切な変換機能がインストールされているからです。

クライアントで ICU が必要になるとき (Windows CE を除くすべてのプラットフォーム)

ユニコードのクライアント・アプリケーションでは、使用するデータベースの文字セットにかかわらず、すべてのクライアントに ICU がインストールされている場合に、クライアントとデータベース・サーバ間のパフォーマンスが向上する可能性があります。これは、必要な変換処理の一部がデータベース・サーバからクライアントに渡されて、負荷が軽減するからです。

また、Windows プラットフォームで ODBC を使用している場合は、ANSI アプリケーションであっても、クライアントに ICU をインストールしておくことが必要です。これは、ドライバ・マネージャが ANSI ODBC 呼び出しをユニコード ODBC 呼び出しに変換するからです。

文字セットに関する質問とその回答

次の表に、各質問とその回答が記述されている参照先を示します。

質問…	参照先…
データベースで使用する照合を決定する方法は？	「照合の知識」 349 ページ
ソフトウェア、特に SQL Anywhere における文字の表示方法は？	「文字セットの知識」 341 ページ
SQL Anywhere が提供する照合の種類は？	「照合の選択」 352 ページ
SQL Anywhere がサポートする文字セット・エンコードは？	「サポートされている文字セット」 359 ページ
クライアント・コンピュータとデータベースとで使用文字セットが違う。クライアントとサーバ間で文字を正しくやり取りする方法は？	「文字セット変換」 344 ページ
接続文字列で使用できる文字セットは？	「接続文字列と文字セット」 344 ページ

質問…	参照先…
既存のデータベースの照合順を変更する方法は？	「データベースの照合を変更する」 357 ページ

文字セットの知識

この項では、国際言語と文字セットに関連した、ソフトウェアの問題の概要について説明します。

文字セット、エンコード、照合の概要

各ソフトウェアは、「**文字セット**」を使用します。文字セットは記号、文字、数字、スペースなどから成ります。"ISO-8859-1" は文字セットの例です。Latin1 とも呼ばれます。

文字を内部的に適切に表すため、各ソフトウェアは「**エンコード**」(「**文字コード**」とも呼ばれる)を使用します。エンコードとは、各文字を 1 バイトまたは複数バイトの情報にマッピングする方法で、16 進数で表します。UTF-8 はエンコードの例です。

「文字セット」と「エンコード」は密接に関連しており、どちらも「エンコード」の意味で使用されることがあります。

「**コード・ページ**」は、エンコードの一形態です。コード・ページとは文字と数値表現とのマッピングのことで、通常、数値表現は 0 ー 255 の整数です。コード・ページの例には、Windows コード・ページ 1252 があります。

このマニュアルでは、「エンコード」、「文字コード」、「文字セット・エンコード」、「コード・ページ」を同じ意味で使用します。

データベース・サーバは、文字をソート (たとえば、名前をアルファベット順にリスト) するときに「**照合**」を使用します。照合は文字コード (文字と表現間のマップ) と文字の「**ソート順**」の組み合わせです。各文字列にソート順が複数ある場合があります。たとえば、大文字／小文字を区別するソート順と大文字／小文字を区別しないソート順があります。また、言語間で同じ文字に対するソート順が異なる場合もあります。

文字は「**フォント**」を使って画面上に表示されます。これは文字セットの文字とその外観との間のマッピングです。フォントはオペレーティング・システムによって処理されます。

オペレーティング・システムは、「**キーボード・マッピング**」を使って、キーボードのキーまたはキーの組み合わせを文字セットの文字にマッピングします。

クライアント／サーバ・コンピューティングにおける言語の問題

クライアント・アプリケーションで作業するデータベース・ユーザは、次のソースから文字列を参照したり、文字列にアクセスしたりする場合があります。

- **データベース内のデータ**　データベースには文字列やその他のテキスト・データが格納されています。データベース・サーバは要求に応答するときに、これらの文字列を処理します。たとえば、データベース・サーバが、テーブルの N より後の文字で始まるすべての名前を表示するよう求められることがあります。この要求では、文字列比較を実行する必要がありますが、特定の文字セットのソート順序が想定されています。

- **データベース・サーバ・ソフトウェア・メッセージ** アプリケーションによってデータベース・エラーが引き起こされることがあります。たとえば、存在しないカラムを参照するクエリをアプリケーションが送信した場合です。この場合、データベース・サーバは警告かエラー・メッセージを返します。このメッセージは「**言語リソース・ライブラリ**」に保持されます。これは SQL Anywhere が使用する DLL または共有ライブラリです。
- **クライアント・アプリケーション** クライアント・アプリケーションのインタフェースはテキストを表示します。また、内部でテキストを処理できます。
- **クライアント・ソフトウェア・メッセージ** クライアント・ライブラリは、データベース・サーバと同じ言語を使用してクライアント・アプリケーションにメッセージを提供します。
- **オペレーティング・システム** クライアントとサーバのオペレーティング・システムは、メッセージを提供したりテキストを処理したりします。

環境を適切に動作させるためには、テキストの入力箇所のすべてで統合的に機能しなければなりません。大まかに言うと、すべてユーザの言語および文字セット、またはそのいずれかで動作させてください。

シングルバイト文字セット

多くの言語では文字の数はシングルバイトの文字セットで扱える程度です。このような文字セットでは、各文字はシングルバイト (2 桁の 16 進数) で表現されます。

シングルバイトのセットでは最大 256 文字を表すことができます。アクセントの付いた文字を含め、国際的に使用するすべての文字を保持できるシングルバイト文字セットはありません。この問題は、1 つ以上の国の言語に適する文字セットを記述するコード・ページのセットの開発により解決されました。たとえば、コード・ページ 1253 にはギリシャ語の文字セットが含まれており、コード・ページ 1252 には西ヨーロッパ言語の文字セットが含まれています。コード・ページは多数あり、その名前も多数あります。上記の例は、Windows のコード・ページです。

上方ページと下方ページ

わずかな例外はありますが、文字 0 ー 127 はすべてのコード・ページで共通です。この範囲の文字のマッピングを「**ASCII**」文字セットと呼びます。これには、英語のアルファベットの大文字と小文字、共通の句読表記記号、数字が含まれます。この範囲は「**7 ビット範囲**」(127 までの文字を表すのに 7 ビットしか必要ないため) または「**下方ページ**」と呼ばれます。128 ー 255 までの文字は「**拡張文字**」、または「**上方コード・ページ文字**」と呼ばれ、コード・ページ間で異なります。

英語のアルファベット文字だけを使用する場合は、各コード・ページの ASCII 部分 (0 ー 127) のみで表せるため、コード・ページの互換性の問題が起こることはほとんどありません。しかし、その他の文字を使用すると、非英語環境ではよく起こることですが、データベースとアプリケーションが異なるコード・ページを使用している場合に、問題が起こる可能性があります。

たとえば、UTF-8 文字セットを使用するデータベースが cp1252 データを含むファイルからテーブルをロードするときに、LOAD TABLE 文でエンコードが cp1252 に指定されていなかったとします。エンコードが指定されていないので、データのエンコードは UTF-8 であると想定され、文字変換は実行されないので、cp1252 エンコードがデータベースに直接格納されます。つま

り、cp1252 では 16 進数の 80 で表されているユーロ記号が UTF-8 に変換されません。UTF-8 のユーロ記号は E2 82 AC の 3 バイト・シーケンスで表されますが、この例の場合はデータベースに 80 として格納されます。後でアプリケーションからデータを要求されたとき、データベース・サーバはデータを UTF-8 からクライアントの文字セットに変換しようとします。この変換により、文字の破損が発生します。

マルチバイト文字セット

言語によっては (日本語や中国語など)、256 文字よりもはるかに多い文字があります。この場合はシングルバイトを使用しては表示できないので、マルチバイトのエンコードを使用する必要があります。さらに、多くの言語の文字を単一の文字セットで表現するために、マルチバイトの文字セットよりも多くの文字を使う文字セットも存在します。この例として、UTF-8 が挙げられます。

マルチバイト文字セットは「**可変幅**」で、いくつかの文字はシングルバイト、他はダブルバイトなどになります。

マルチバイト文字セットと照合の詳細については、「SQL Anywhere 照合アルゴリズム (SACA)」 349 ページを参照してください。

例

たとえば、コード・ページ 932 (日本語) の文字の長さは 1 バイトまたは 2 バイトです。最初のバイト (「リード・バイト」とも呼ばれる) の値が 16 進数値 ¥x81 ー ¥x9F または ¥xE0 ー ¥xEF (10 進数値 129 ー 159 または 224 ー 239) の範囲にある場合、その文字は 2 バイト文字であり、直後のバイト (「フォロー・バイト」とも呼ばれる) と併せて文字が成立します。フォロー・バイトとは、最初のバイト以外のすべてのバイトのことです。

最初のバイトがリード・バイトの範囲外にある場合、その文字はシングルバイト文字であり、次のバイトは次の文字の最初のバイトになります。

Windows 環境の ANSI コード・ページと OEM コード・ページ

Windows の場合、2 つのコード・ページが使用されています。Windows グラフィカル・ユーザ・インタフェースを使用するアプリケーションでは、Windows コード・ページが使用されます。Windows コード・ページには、ISO 文字セットおよび ANSI 文字セットとの互換性があります。このようなコード・ページは、しばしば「**ANSI コード・ページ**」と呼ばれます。

Windows で動作する文字モード・アプリケーション (コンソールまたはコマンド・プロンプト・ウィンドウを使用するアプリケーション) は DOS で使用されていたコード・ページを使用します。これは歴史的な理由から「**OEM コード・ページ**」(Original Equipment Manufacturer) と呼ばれます。

SQL Anywhere は、OEM と ANSI コード・ページの両方に基づいた照合をサポートします。OEM 照合は互換性確保のためにサポートされていますが、新しいデータベースでは使用しないでください。「サポートされている照合と代替照合」 360 ページを参照してください。

SQL Anywhere データベース内での文字セット

SQL Anywhere データベースでは、文字データの格納に 1 つまたは 2 つの文字セット (エンコード) を使用できます。CHAR データ型 (CHAR、VARCHAR、LONG VARCHAR など) では、シングルバイト文字セットまたはマルチバイト文字セットが使用されます。UTF-8 も使用できます。NCHAR データ型 (NCHAR、NVARCHAR、LONG NVARCHAR など) では、UTF-8 が使用されます。

CHAR データ型と NCHAR データ型の詳細については、「文字データ型」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

文字セット変換

SQL Anywhere は、文字セットまたはコード・ページの異なる場所で同じ文字を表している文字セット間で、文字セット変換を実行できます。これを可能にするには、文字セット間にある程度の互換性が必要になります。たとえば、EUC-JIS と cp932 との間では文字セット変換を実行できますが、EUC-JIS と cp1252 との間では実行できません。

SQL Anywhere では、IBM が開発および保守している International Components for Unicode (ICU) オープン・ソース・ライブラリを使用して文字セット変換を実装してます。

異なるデータ型の値を比較するための文字セット変換の詳細については、「データ型間の比較」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

接続文字列と文字セット

使用している文字セットがクライアント間で異なる場合、接続文字列の文字セット変換はたいへん困難になります。接続文字列は、データベース・サーバを検出または起動するために、クライアント・ライブラリによって解析されますが、解析はデータベース・サーバの文字セットや言語が未知のまま実行されます。

インタフェース・ライブラリは、接続文字列を次のように解析します。

1. 接続文字列は、*keyword=value* の組み合わせに分解されます。これは、CommLinks (LINKS) パラメータの前後に中カッコ {} を使用しないかぎり、文字セットにかかわらず実行されます。中カッコの代わりにカッコ () を使用することをおすすめします。中カッコは、一部のマルチバイト文字セットで有効な「**フォロー・バイト**」(最初のバイト以外のバイト) です。
2. サーバが検出されます。サーバ名に対して文字セット変換は実行されません。クライアントとデータベース・サーバとで文字セットが異なる場合、サーバ名に拡張文字が使用されているとサーバが見つからないことがあります。

   コンピュータ間の互換性を最大限確保するため、サーバ名には英数字 (ASCII 文字 32 ー 126) を使用してください。
3. DatabaseName (DBN) 接続パラメータまたは DatabaseFile (DBF) 接続パラメータは、クライアントの文字セットからデータベース・サーバの文字セットに変換されます。

4. データベースが検出されると、残りの接続パラメータがデータベースの文字セットに変換されます。

外国語での大文字と小文字の区別

SQL Anywhere では通常、システム・ビュー名やカラム名などの識別子の「**大文字と小文字を維持**」し、「**大文字と小文字を区別しません**」。名前は作成時の大文字と小文字のまま格納されますが、識別子へのアクセスは大文字と小文字の区別なしで行われます。

たとえば、システム・ビューの名前は大文字 (SYSDOMAIN、SYSTAB など) で格納されますが、アクセスは大文字と小文字の区別なく行うことができます。したがって、次の 2 つの文は同等です。

```
SELECT * FROM systab;
SELECT * FROM SYSTAB;
```

照合では、大文字と小文字が同等と定義されています。ただし、一部の照合では、識別子の大文字と小文字の区別を前提とする場合、特別な注意が必要です。たとえば、トルコ語の照合では、予期できない複雑なエラーが発生するような大文字と小文字の変換動作があります。最も一般的なエラーは、**I** または **i** という文字を含むシステム・オブジェクトが見つからないというものです。

トルコ語文字セットと照合の詳細については、「トルコ語文字セットと照合」 364 ページを参照してください。

ロケールの知識

データベース・サーバとクライアント・ライブラリはどちらも、「**ロケール定義**」を使用して、言語と文字セット環境を認識します。

ロケールの概要

アプリケーションのロケールまたはクライアントのロケールは、データベース・サーバへの要求時にクライアントまたはクライアント・ライブラリによって使用され、返す結果に使用される文字セットとともに、エラー・メッセージ、警告、その他のメッセージの言語を決定します。データベース・サーバは自身のロケールをアプリケーションのロケールと比較し、必要な文字セット変換を判断します。サーバ上のデータベースによって、ロケール定義が異なる場合があります。また、クライアントによってロケールが異なる場合もあります。

ロケールは次のコンポーネントで構成されています。

- **言語**　ISO-639 規格値を使用した 2 文字の文字列です。ドイツ語は DE、フランス語は FR などのように表現されます。データベース・サーバとクライアントのどちらにも、自身のロケールに対する言語の値があります。

  データベース・サーバは、ロードする言語ライブラリをロケール言語で判断します。データベースの作成時に照合が指定されていな場合、データベース・サーバは使用する照合の判断に文字セットと併せてロケール言語も使用します。

  クライアント・ライブラリでは、ロードする言語ライブラリをロケール言語で判断し、データベースからの要求もロケール言語で判断します。「ロケール言語の知識」 346 ページを参照してください。

- **文字セット**　文字セットとは、使用しているコード・ページまたはエンコードのことです。クライアントとサーバのどちらにも文字セットの値がありますが、両者が異なる場合もあります。異なる場合は、文字セット変換を実行して相互運用性を確保します。「ロケール文字セットの知識」 348 ページを参照してください。

ロケール言語の知識

ロケール言語は、クライアント・アプリケーションのユーザによって使用される言語、またはデータベース・サーバのユーザによって使用されることが予測される言語です。ロケール設定の検索方法については、「ロケール情報の確認」 354 ページを参照してください。

クライアント・ライブラリとデータベース・サーバはどちらも、ロケールの言語コンポーネントを同じ方法で確認します。

1. SALANG 環境変数が指定されている場合は、その値を使用します。「SALANG 環境変数」 313 ページを参照してください。

2. Windows において、SALANG 環境変数に値の指定がない場合、SQL Anywhere 言語レジストリのエントリを確認します。「インストール時のレジストリ設定」 329 ページを参照してください。
3. オペレーティング・システムの言語設定の確認
4. 上記の設定で言語を判断できなかった場合は、デフォルトで英語になります。

言語ラベルの値

次の表は、有効な言語ラベルの値と対応する ISO 639 言語コードを示します。

言語	ISO_639 言語コード	言語ラベル	代替ラベル
アラビア語	AR	arabic	なし
チェコ語	CS	czech	なし
デンマーク語	DA	danish	なし
オランダ語	NL	dutch	なし
英語	EN	us_english	english
フィンランド語	FI	finnish	なし
フランス語	FR	french	なし
ドイツ語	DE	german	なし
ギリシャ語	EL	greek	なし
ヘブライ語	HE	hebrew	なし
ハンガリー語	HU	hungarian	なし
イタリア語	IT	italian	なし
日本語	JA	japanese	なし
韓国語	KO	korean	なし
リトアニア語	LT	lithuanian	なし
ノルウェー語	NO	norwegian	norweg
ポーランド語	PL	polish	なし
ポルトガル語	PT	portuguese	portugue
ロシア語	RU	russian	なし
中国語 (簡体文字)	ZH	chinese	simpchin

言語	ISO_639 言語コード	言語ラベル	代替ラベル
スペイン語	ES	spanish	なし
スウェーデン語	SV	swedish	なし
タイ語	TH	thai	なし
中国語 (繁体文字)	TW	tchinese	tradchin
トルコ語	TR	turkish	なし
ウクライナ語	UK	ukrainian	なし

ロケール文字セットの知識

アプリケーションとサーバのロケール定義のいずれにも文字セットがあります。アプリケーションは、データベース・サーバから文字列を要求するときに文字セットを使用します。データベース・サーバはデータベース文字セットをアプリケーションの文字セットと比較し、必要な文字セット変換を確認します。データベース・サーバがクライアントの文字セットを変換できない場合は、接続が失敗します。

1. SACHARSET 環境変数が設定されている場合、その値が文字セットの確認に使用されます。「SACHARSET 環境変数」 311 ページを参照してください。

   データベース・サーバが SACHARSET を使用するのは、照合の指定がない状態で新しいデータベースを作成する場合だけです。

2. 接続文字列で文字セットが指定されている場合は、その文字セットが使用されます。詳細については、「CharSet 接続パラメータ [CS]」 234 ページを参照してください。

3. Open Client アプリケーションは、*Sybase* リリース・ディレクトリの *locales* サブディレクトリの *locales.dat* ファイルを確認します。

4. オペレーティング・システムからの文字セット情報を使用して、ロケールが次のように決定されます。

   - Windows オペレーティング・システムの場合、最新の Windows ANSI コード・ページが使用されています。
   - UNIX プラットフォームの場合、次のロケール環境変数が LC_ALL、LC_MESSAGES、LC_CTYPE、LANG の順序で確認されます。この中で最初に設定されていた環境変数の値が、文字セットの確認に使用されます。文字セットをオペレーティング・システムから確認できなかった場合は、デフォルトで iso_1 (Windows コード・ページ 28591、ISO 8859-1 Latin I、ISO 8859-1 Latin-1、iso_8859-1:1987 とも呼ばれる) が使用されます。

5. それ以外のプラットフォームでは、デフォルトでコード・ページ cp1252 が使用されます。

ロケール設定の検索方法については、「ロケール情報の確認」 354 ページを参照してください。

照合の知識

照合とは、特定の文字セットまたはエンコードに対する文字のソートおよび比較の方法です。SQL Anywhere では、照合アルゴリズムとして、SQL Anywhere 照合アルゴリズム (SACA) とユニコード照合アルゴリズム (UCA) の 2 つをサポートしています。SACA を使用すると、ソートが高速、簡潔、実用的になりますが、言語的な正確さが低下します。UCA を使用すると、言語的な処理は正確になりますが、記憶領域の要件と実行時間が多少増加します。

この項では、提供される照合と、状況に応じた照合の使い分けについて説明します。

特定の照合を使用してデータベースを作成する方法については、「名前を付けた照合を使用してデータベースを作成する」 356 ページと「初期化ユーティリティ (dbinit)」 662 ページを参照してください。

SQL Anywhere 照合アルゴリズム (SACA)

SQL Anywhere 照合アルゴリズムは、シングルバイト文字とマルチバイト文字の実用的な比較、ソート、大文字小文字変換を提供します。このアルゴリズムは、必要とする領域が少なく、高速です。インデックスなどの文字列は、マッピング後も元の文字列と長さが同じです。比較、ソート、大文字小文字変換のマッピングでは、文字列の各バイト値ごとの簡単なテーブル・ルックアップを使用しています。

SACA は、初期の Watcom SQL 以来、SQL Anywhere に提供され続けています。

シングルバイト文字セット

典型的なシングルバイト文字セットの照合では、各文字のアクセント記号付きとアクセント記号なしのあらゆる形が同じ値にマッピングされており、照合ではアクセント記号は区別されません。同じ文字のアクセント記号付き形とアクセント記号なし形は同じとみなされ、ソートされると隣り合わせになります。

この照合では、アクセント記号を維持したまま、大文字と小文字が変換されます。

マルチバイト文字セット

マルチバイト文字セットの場合、リード・バイトが 256 個の異なる値にマッピングされています。フォロー・バイトはバイナリ値として比較されます。

マルチバイト文字セットのほとんどの照合では、文字セット・エンコードにより、文字はリード・バイトによって識別される 256 バイト単位のページにグループ化されるので、このマッピング方法で実用的なソート結果が得られます。これらのページと各ページに含まれる文字は、該当する文字セットとして合理的な順序で配置されています。通常、この照合では文字セットにおけるページの順序 (リード・バイト) が保持されます。ページによっては、他の特徴に基づいてソートされます。たとえば、日本語コード・ページ 932 用の 932JPN 照合では、全角文字 (漢字) と半角文字 (カタカナ) をグループ分けしています。

大文字小文字変換は、7 ビットの英文字についてのみ提供されています。

UTF-8 文字セット

UTF-8 はマルチバイト文字セットです。各文字は 1 ー 4 バイトで構成されます。SQL Anywhere では、UTF-8 文字のソート用に UTF8BIN 照合を提供しています。

UTF8BIN では、リード・バイトが 256 の異なる値にマッピングされており、フォロー・バイトはバイナリ値として比較されます。UTF-8 の文字表現方法と 256 というマッピングの数的制限に起因して、同じ文字のアクセント記号付き形とアクセント記号なし形など、関連文字をグループ化できません。ソートの基準は基本的にバイナリです。

大文字小文字変換は、7 ビットの英文字についてのみサポートされています。

ユニコード照合アルゴリズム (UCA)

ユニコード照合アルゴリズム (UCA) は、ユニコード文字セット全体のソートに使用するアルゴリズムです。これにより、言語的に正しい比較、ソート、大文字小文字変換が実現されます。UCA はユニコード規格の一部として開発されました。SQL Anywhere では、IBM が開発および保守している International Components for Unicode (ICU) オープン・ソース・ライブラリを使用して UCA を実装しています。

注意
デフォルトの UCA ソート順により、ほとんどの言語のほとんどの文字が適切な順序でソートされます。ただし、同じ文字を使用する言語間でソートや比較にさまざまな違いがあるため、UCA ですべての言語について適切なソート順が得られるわけではありません。そのため、ICU は UCA を調整できる構文を提供しています。

UCA を使用すると、少ない領域と時間で高度な比較、ソート、大文字小文字変換を実現できます。

マッピング後の文字列は元の文字列より長くなります。このアルゴリズムは、複雑な文字を的確に処理できます。

SQL Anywhere 照合アルゴリズムとは異なり、ユニコード照合アルゴリズムはシングルバイトの UTF-8 文字セットにのみ使用され、各文字を 1 つまたは複数の属性で区別します。文字の場合、属性は基底文字、アクセント、大文字小文字です。

文字以外の場合、通常は基底文字だけが属性になります。

UCA は、次の方法で文字を比較します。

- 基底文字を比較します。文字列の基底文字が他の文字列と異なる場合は、その時点で比較が完了します。アクセント記号や大文字小文字の違いは考慮されません。
- データベースでアクセント記号の違いが区別される場合は、アクセント記号が比較されます。アクセント記号が異なる場合、その時点で比較が完了します。大文字小文字の違いは考慮されません。
- データベースで大文字と小文字が区別される場合は、各文字の大文字と小文字が比較されます。

元の文字列の値が同じとみなされるのは、基底文字、アクセント記号、大文字小文字がまったく同じ場合だけです。

例

UCA を使用して次の表の第 1 カラムの文字列を比較するとします。後続のカラムには各文字列の 3 つの属性が記載されています。基底文字は同じで、違うのはアクセント記号と大文字小文字だけです。

文字列	基底文字	アクセント記号	大文字小文字
noel	noel	なし、なし、なし、なし	小、小、小、小
noel	noel	なし、なし、アクセント、なし、	小、小、小、小
Noel	noel	なし、なし、なし、なし	大、小、小、小
Noel	noel	なし、なし、アクセント、なし、	大、小、小、小

次の表は、UCA を使用した場合にアクセント記号と大文字小文字の区別により可能な 4 つの条件によるソート順を示します。

アクセント記号の区別	[大文字と小文字を区別]	ORDER BY による結果	説明
N	N	Noel、noel、Noel、noel (順不同)	◆ アクセント記号は区別しない ◆ 大文字と小文字も区別しない ◆ すべての値が同じとみなされる ◆ 順序は要素数 4 の集合内でランダム
Y	N	Noel、noel (順不同) 次の文字が継続 noel、Noel (順不同)	◆ アクセント記号なしが先、アクセント記号ありが後。したがって e の前に e がくる ◆ 大文字と小文字は区別しない。N と n は 2 つの間でランダム
N	Y	Noel、Noel (順不同) 次の文字が継続 noel、noel (順不同)	◆ 大文字が先、小文字が後。したがって n の前に N がくる ◆ アクセント記号は区別しない。e と e は 2 つの間でランダム
Y	Y	Noel noel Noel noel	◆ アクセント記号なしが先、アクセント記号ありが後。したがって e の前に e がくる ◆ 大文字が先、小文字が後。したがって n の前に N がくる

SQL Anywhere データベースでの照合

CHAR 照合

CHAR データ型 (CHAR、VARCHAR、LONG VARCHAR など) の照合では、SQL Anywhere 照合アルゴリズムが使用される場合とユニコード照合アルゴリズムが使用される場合があります。どちらの場合も、使用される照合は CHAR 照合と呼ばれます。

NCHAR 照合

NCHAR データ型 (NCHAR、NVARCHAR、LONG NVARCHAR など) の照合では、ユニコード照合アルゴリズムが使用される場合と UTF8BIN 照合が使用される場合があります。UTF8BIN 照合では SQL Anywhere 照合アルゴリズムが使用されます。

大文字小文字とアクセント記号の区別

作成時に大文字小文字を区別するよう指定されなかった SQL Anywhere データベースでは、大文字と小文字は区別されません。区別されるようにするには、該当するオプションを指定します。データベースの作成後、データベースを再構築しないで大文字と小文字が区別されるようにすることはできません。

データベースが大文字と小文字を区別するかどうかによって、SACA 照合でも UCA 照合でも大文字と小文字を区別するかどうかが決まり、それにより CHAR 照合と NCHAR 照合で大文字と小文字を区別するかどうかが決まります。

作成時にアクセント記号を区別するよう指定されなかった SQL Anywhere データベースでは、アクセント記号は区別されません。区別されるようにするには、該当するオプションを指定します。データベースの作成後、データベースを再構築しないでアクセント記号が区別されるようにすることはできません。

データベースがアクセント記号を区別するかどうかは、UCA 照合にのみ影響を与えます。UCA 照合が CHAR 照合や NCHAR 照合で使用されているかどうかには関係ありません。 CHAR 照合と NCHAR 照合のどちらにも SACA 照合を使用することを選択した場合、アクセント記号に関するオプションに効力はありません。アクセント記号の区別は SACA 照合の属性であり、該当するオプションをデータベース作成時に使用して指定することはできません。

照合の選択

データベースを作成するとき、SQL Anywhere ではオペレーティング・システムの言語と文字セットの設定に基づいてデフォルト照合を選択できます。ほとんどの場合、デフォルト照合は適切な選択ですが、用意されている多数の照合の中からニーズに合った照合を明示的に選択することもできます。SQL Anywhere が特定の言語に対して複数の照合をサポートする場合もあります。

データベースのデータに適した文字セットとソート順を使用する照合を選択してください。文字列のソートや比較を詳細に制御することを目的に、照合の適合化オプションを指定することもできます。データベースの作成については、「データベースの作成」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

データのソートと国際化機能の詳細については、「SQL Anywhere の国際化機能」 336 ページを参照してください。

参照

- 「チュートリアル：SQL Anywhere データベースの作成」『SQL Anywhere サーバ - SQL の使用法』
- 「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「初期化ユーティリティ (dbinit)」 662 ページ

照合の選択時の考慮事項

使用するデータベース用の照合を選択するには、以下について考慮します。

- 文字セット変換を使用する場合は、必要以上にシステム設定が複雑になるだけでなく、パフォーマンスも犠牲になります。このため、文字セット変換の必要がない照合を選択してください。データベース・サーバとクライアントが同じ文字セットを使用している場合は、文字セット変換は使用されません。

  文字セット変換を回避するには、クライアント・コンピュータのオペレーティング・システムで使用されている文字セットに対応する照合順をデータベースで使用します。クライアント・コンピュータのオペレーティング・システムが Windows の場合は、ANSI 文字セットを選択してください。

- クライアント・コンピュータでさまざまな文字セットを扱う場合、またはデータベースにユニコード・データを格納する必要がある場合、UCA 照合か UTF8BIN 照合またはその両方を使用することを検討します。ただし、UTF-8 以外のマルチバイト文字セットに対しては UCA 照合を使用する必要があることに注意してください。

- データベースのデータに適した文字セットとソート順を使用する照合を選択します。この条件を満たす照合が複数ある場合もあります。

SQL Anywhere による新しいデータベースのデフォルトの照合の選択方法

新しいデータベースが作成され、照合が明示的に指定されていない場合、SQL Anywhere では言語と文字セットを使用して照合を決定します。

- 言語は、SALANG 環境変数 (存在する場合)、レジストリ、オペレーティング・システムで判断します。「SALANG 環境変数」 313 ページを参照してください。

- 文字セットは、SACHARSET 環境変数 (存在する場合) またはオペレーティング・システムで判断します。「SACHARSET 環境変数」 311 ページを参照してください。

国際言語と文字セットのタスク

この項では、国際言語と文字セットの問題に関連したタスクについて、まとめて説明します。

デフォルトの照合の判断

データベースの作成時に照合を指定しないと、デフォルトの照合が使用されます。デフォルトの照合は、使用するオペレーティング・システムによって異なります。

♦ **使用コンピュータのデフォルトの照合を判断するには、次の手順に従います。**

1. Interactive SQL を起動します。サンプル・データベースに接続します。
2. 次のクエリを入力します。

   ```
   SELECT PROPERTY( 'DefaultCollation' );
   ```

   デフォルトの照合が返されます。

   照合の詳細については、「照合の選択」 352 ページを参照してください。

ロケール情報の確認

ロケール情報は、PROPERTY、DB_PROPERTY、CONNECTION_PROPERTY などの関数を使用して確認できます。次の表は、これらの関数を使用して、クライアント接続、データベース、データベース・サーバのロケール情報を返す方法を示します。

システム関数とパラメータ	戻り値
SELECT PROPERTY('CharSet');	データベース・サーバの文字セット。通常は、サーバを実行しているコンピュータの文字セット
SELECT PROPERTY('DefaultCollation');	データベース・サーバがデータベース作成時に使用するデフォルトの CHAR 照合
SELECT PROPERTY('DefaultNcharCollation');	データベース・サーバがデータベース作成時に使用するデフォルトの NCHAR 照合
SELECT PROPERTY('Language');	サーバ・コンソールが使用する言語
SELECT DB_PROPERTY('CharSet');	CHAR データをデータベースに格納するときに使用する文字セット
SELECT DB_PROPERTY('NcharCharSet');	NCHAR データをデータベースに格納するときに使用する文字セット

システム関数とパラメータ	戻り値
SELECT DB_PROPERTY('MultiByteCharSet');	CHAR データでマルチバイト文字セットを使用するかどうか (On=使用、Off=使用しない)
SELECT DB_PROPERTY('Language');	データベースの CHAR 照合でサポートしている言語を表す 2 文字のコードをカンマで区切ったリスト
SELECT DB_PROPERTY('Collation');	データベース・サーバが使用する CHAR 照合名
SELECT DB_PROPERTY('NcharCollation');	データベース・サーバが使用する NCHAR 照合名
SELECT CONNECTION_PROPERTY('CharSet');	クライアントの CHAR データ文字セット
SELECT CONNECTION_PROPERTY ('NcharCharSet');	接続に使用する NCHAR データの文字セット
SELECT CONNECTION_PROPERTY ('Language');	クライアントが接続に使用する言語

参照

- 「PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』
- 「DB_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』
- 「CONNECTION_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』

ロケールの設定

オペレーティング・システムのデフォルトのロケールを使用するか、コンピュータの SQL Anywhere コンポーネントで使用するロケールを明示的に設定できます。

♦ **SQL Anywhere のロケールを設定するには、次の手順に従います。**

1. デフォルトのロケールで問題ない場合は、何の作業も必要ありません。

   オペレーティング・システムのデフォルト・ロケールを検索する方法については、「ロケール情報の確認」 354 ページを参照してください。

2. ロケールを変更する必要がある場合は、SALANG 環境変数か SACHARSET 環境変数のまたはその両方を設定します。

   ```
   SACHARSET=charset
   SALANG=language_code
   ```

   charset は有効な文字セット・ラベルで、*language_code* は有効な言語のリストにある言語コードです。「言語ラベルの値」 347 ページを参照してください。

各種オペレーティング・システムで環境変数を設定する方法については、「SQL Anywhere の環境変数」 305 ページを参照してください。

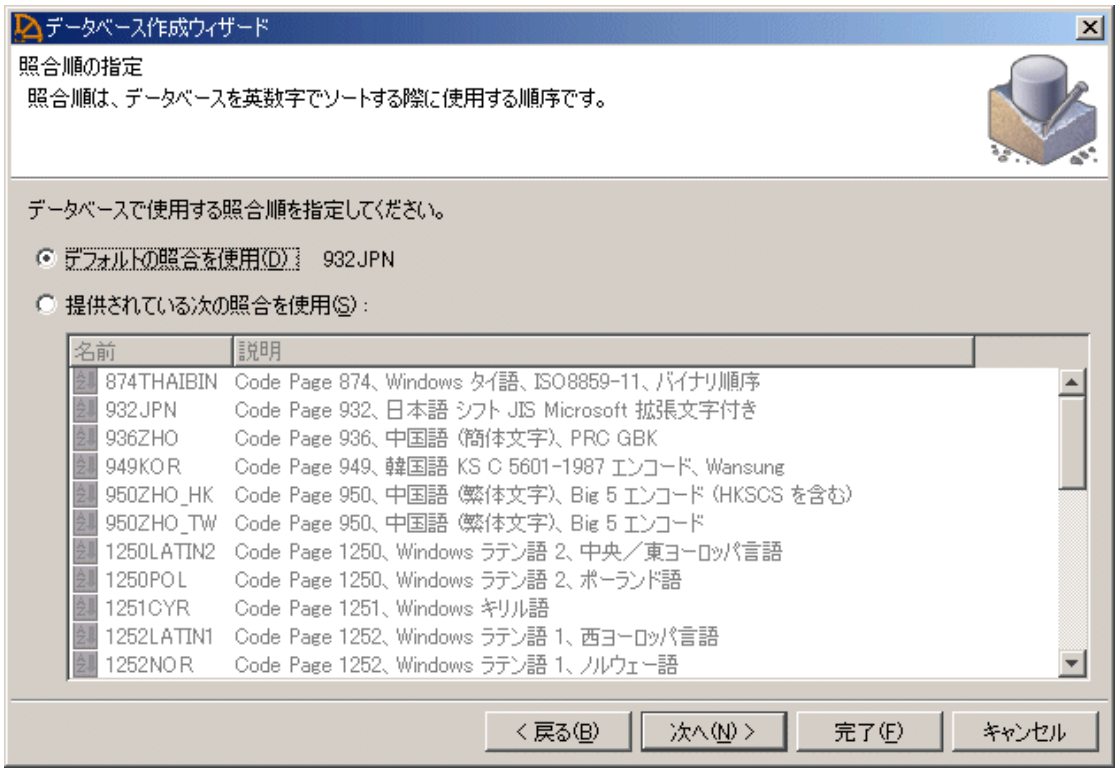
名前を付けた照合を使用してデータベースを作成する

データベースの作成時に、各データベースで使用する照合を指定できます。デフォルトの照合は、データベース・サーバのオペレーティング・システムで使用されるコード・ページと言語から推定されます。

NCHAR 照合の使用方法の詳細については、「NCHAR 照合」 352 ページを参照してください。

♦ データベース作成時にデータベース照合を指定するには、次の手順に従います (Sybase Central の場合)。

- Sybase Central では、[データベース作成] ウィザードを使用してデータベースを作成できます。ウィザードの中に、リストから照合を選択するページがあります。



♦ データベース作成時にデータベース照合を指定するには、次の手順に従います (コマンド・プロンプトを使用する場合)。

1. コマンド・プロンプトで次のように入力して、推奨する照合順をリストします。

   ```
   dbinit -l
   ```

リストの最初のカラムは照合ラベルです。これは、データベースの作成時に指定します。

2. dbinit ユーティリティを使用して -z オプションで照合順を指定し、データベースを作成します。次のコマンドは、ギリシャ語の照合を設定したデータベースを作成します。

```
dbinit -z 1253ELL mydb.db
```

♦ **データベース作成時にデータベース照合を指定するには、次の手順に従います (SQL の場合)。**

· CREATE DATABASE 文を使用して、データベースを作成できます。次の文は、ギリシャ語の照合を設定したデータベースを作成します。

```
CREATE DATABASE 'mydb.db'
COLLATION '1253ELL'
```

参照

- 「チュートリアル：SQL Anywhere データベースの作成」『SQL Anywhere サーバ - SQL の使用法』
- 「データベースの作成」『SQL Anywhere サーバ - SQL の使用法』
- 「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「初期化ユーティリティ (dbinit)」 662 ページ

データベースの照合を変更する

データベースの照合を変更するには、データベースを再構築する必要があります。照合はデータベースの作成時に選択されるものであり、変更はできません。

♦ **照合を変更するには、次の手順に従います。**

1. 既存のデータベースの文字セットを次のコマンドで確認します。

```
SELECT DB_PROPERTY( 'CharSet' );
```

SQL Anywhere の初期バージョンでは、このプロパティがない場合があります。文字セットは照合名で判断することもできます。たとえば、照合 1252LATIN1 はコード・ページ 1252 を使用しています。

2. 既存のデータベースに格納されているデータの文字セットを確認します。

本来はデータベースの文字セットと同じはずですが、異なっている場合は、データベースを再構築するよい機会です。ただし、再構築の実施には十分な注意が求められます。

特に、使用しているデータベースの照合が 850LATIN1 であり、使用している SQL Anywhere が初期バージョンであるため文字セット変換がサポートされていないか (バージョン 5 以前) デフォルトで無効になっており (バージョン 6 と 7)、クライアント・アプリケーションが標準的な Windows アプリケーションの場合、データベースにコード・ページ 1252 の文字データが含まれている可能性があります (通常はコード・ページ 850 に含まれる文字データです)。このケースに該当するかどうかを簡単にテストするには、UNLOAD TABLE に ENCODING オプションを指定して文字データの一部をアンロードし、Windows のメモ帳で表示します。アクセント記号付きデータが正しい場合、データベースに含まれている文字

データは Windows ANSI コード・ページに対応しており、英語と西ヨーロッパ言語の場合はコード・ページ 1252 です。データが DOS ベースのエディタで正常に表示される場合、文字データは Windows OEM コード・ページに対応しており、通常は 437 または 850 です。

3. データベースをアンロードします。

   データの文字セットがデータベースの文字セットと互換性がない場合、文字セット変換なしでデータをアンロードすることが重要です。使用されている SQL Anywhere によっては、dbunload の内部アンロード機能を使用したり、UNLOAD TABLE 文を使用してデータを手動でアンロードしたりできます。

4. 新しいデータベースを作成し、使用する照合と文字セットを指定します。

5. データを新しいデータベースにロードします。

   アンロードしたデータとスキーマ (*reload.sql*) が再ロードに使用するコンピュータの文字セットに対応している場合、dbunload の外部再ロード・オプションを使用できます。データは、サーバの文字セット変換により、データベースの正しい文字セットに自動変換されます。

   データのエンコードがデータベースの文字セットと一致してなく、データのロードに LOAD TABLE 文 (内部再ロード) を使用している場合は、ENCODING 句を使用する必要があります。データベース・サーバは、LOAD TABLE 文を使用してロードされたデータに対し、デフォルトでは文字セット変換を実行しません。

   データのエンコードが作業に使用しているコンピュータのコード・ページと一致せず、ロードに INPUT 文 (外部再ロード) を使用している場合は、ENCODING 句を使用する必要があります。そうしないと、データベース・サーバはデータがコンピュータのネイティブ文字セットであると想定します。

参照

- 「LOAD TABLE 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「UNLOAD TABLE 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「INPUT 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「名前を付けた照合を使用してデータベースを作成する」 356 ページ
- 「アンロード・ユーティリティ (dbunload)」 739 ページ
- 「データベースの再構築」 『SQL Anywhere サーバ - SQL の使用法』
- 「CharSet 接続パラメータ [CS]」 234 ページ

文字セットと照合の参考情報

以降の項では、SQL Anywhere の文字セットと照合に関する情報を説明します。

サポートされている文字セット

SQL Anywhere では、数多くの文字セットとラベルをサポートしており、その対象は増え続けています。文字セット・エンコード・ラベルは、さまざまな名前やラベルが知られています。SQL Anywhere でサポートされている文字セットのリストを表示するには、コマンド・プロンプトで次のコマンドを実行します。

```
dbinit -le
```

出力の各行には、指定した文字セット・エンコードの代替ラベルがカンマで区切られてリストされます。各行の最初のラベルは、その文字セット・エンコードで優先される SQL Anywhere 名です。その他のラベルは、別の機関、組織、または規格によって使用されるもので、IANA (Internet Assigned Numbers Authority)、MIME (Multipurpose Internet Mail Extensions)、ICU (International Components for Unicode)、JAVA、および ASE (Adaptive Server Enterprise) があります。

目的の文字セットが見つからない場合は、次のコマンドを実行して、一般には使用されていない (または知られていない) 文字セットを含むリストを参照できます。

```
dbinit -le+
```

文字セット・エンコードのラベルを指定すると、SQL Anywhere によってラベルの検索が行われます。場合によっては、他の機関が異なる文字セットに対して同じラベルを使用することがあります。SQL Anywhere は、可能なかぎり、あいまいさを解決しようとします。たとえば、あいまいなラベルで設定されている文字を参照する JDBC アプリケーションは、JAVA 規格のラベルに対して解決が試行されます。あいまいさを回避するため、常に SQL Anywhere ラベルを使用することをおすすめします。文字セット・エンコードのラベルへの理解を深めるための優れた資料として、「International Components for Unicode」があります。

特定の文字セットまたはラベルがサポートされているかどうかを確認する別の方法として、CSCONVERT 機能を使用できます。CSCONVERT 機能は、データベース文字セットによる文字列を、指定した文字セットまたはラベルに変換します。文字セットまたはラベルがサポートされていない場合、データベース・サーバはエラーを返します。文字セットまたはラベルがサポートされている場合、データベース・サーバは結果セットを返します。

たとえば、cp850 文字セットがサポートされているかどうかを確認するには、Interactive SQL で次のコマンドを実行します。

```
SELECT CSCONVERT ( 'teststring', 'cp850' );
```

データベース・サーバは結果の値を返します。これは、cp850 がサポートされていることを示しています。

CSCONVERT 関数を使用する場合の詳細については、「CSCONVERT 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

その他の文字セット・エンコード・ラベル

SQL Anywhere では、さまざまな文字セット・エンコードのラベルが認識されます。SQL Anywhere、IANA、MIME、ICU、JAVA、または ASE などの標準的なラベルではないラベルもあります。サポートされている文字セット・エンコードのラベルのリストを表示するには、コマンド・プロンプトで次のコマンドを実行します。

```
dbinit -le+
```

出力の各行には、指定した文字セット・エンコードの代替ラベル全体がリストされます。各行の最初のラベルは、その文字セット・エンコードで優先される SQL Anywhere 名です。

サポートされている照合と代替照合

次のリストは、SQL Anywhere でサポートされている CHAR 照合を示します。このリストは、コマンド・プロンプトで次のコマンドを実行しても得られます。

```
dbinit -l
```

照合ラベル	説明
874THAIBIN	Code Page 874、Windows Thai、ISO8859-11、バイナリ順序
932JPN	Code Page 932、日本語シフト JIS、Microsoft 拡張文字付き
936ZHO	Code Page 936、中国語 (簡体文字)、PRC GBK 2312-80 8 ビット・コード
949KOR	Code Page 949、韓国語 KS C 5601-1987 コード、完成型
950ZHO_HK	Code Page 950、中国語 (繁体文字)、Big 5 コード (HKSCS を含む)
950ZHO_TW	Code Page 950、中国語 (繁体文字)、Big 5 コード
1250LATIN2	Code Page 1250、Windows ラテン語 2、中央／東ヨーロッパ言語
1250POL	Code Page 1250、Windows ラテン語 2、ポーランド語
1251CYR	Code Page 1251、キリル語
1252LATIN1	Code Page 1252、Windows ラテン語 1、西ヨーロッパ言語
1252NOR	Code Page 1252、Windows ラテン語 1、ノルウェー語
1252SPA	Code Page 1252、Windows ラテン語 1、スペイン語
1252SWEFIN	Code Page 1252、Windows ラテン語 1、スウェーデン語／フィンランド語
1253ELL	Code Page 1253、Windows ギリシア語、ISO8859-7 拡張付き

照合ラベル	説明
1254TRK	Code Page 1254 Windows ラテン語 5、トルコ語、ISO 8859-9 拡張付き
1254TRKALT	Code Page 1254、Windows トルコ語、拡張付き ISO8859-9、I-dot と I-no-dot は同じ
1255HEB	Code Page 1255、Windows ヘブライ語、ISO8859-8 拡張付き
1256ARA	Code Page 1256、Windows アラビア語、ISO8859-6 拡張付き
1257LIT	Code Page 1257、リトアニア語
EUC_CHINA	中国語 (簡体文字) の GB 2312-80 コード
EUC_JAPAN	日本語の EUC JIS X 0208-1990 と JIS X 0212-1990 コード
EUC_KOREA	韓国語の KS C 5601-1992 コード、Johab (組合型)
EUC_TAIWAN	台湾語の Big 5 コード
ISO1LATIN1	ISO8859-1、ISO ラテン語 1、西ヨーロッパ言語、ラテン語 1 順序
ISO9LATIN1	ISO8859-15、ISO ラテン語 9、西ヨーロッパ言語、ラテン語 1 順序
ISO_1	ISO8859-1、ラテン語 1、西ヨーロッパ言語
ISO_BINENG	バイナリ順序、英語 ISO/ASCII 7 ビット文字ケース・マッピング
UCA	標準のデフォルト UCA 照合
UTF8BIN	UTF-8、ユニコード用 8 ビット・マルチバイト・エンコード、バイナリ順序

代替照合

代替照合は、古いバージョンの SQL Anywhere との互換性や、その他の特殊な用途で使用できます。サポートされている代替照合の一覧を表示するには、コマンド・プロンプトで次のコマンドを実行します。

```
dbinit -l+
```

参照

- 「初期化ユーティリティ (dbinit)」 662 ページ

推奨文字セットと照合

SQL Anywhere は数多くの文字セット、コード・ページ、エンコード、照合の名前を認識しますが、この項では Windows プラットフォームと UNIX プラットフォームで使用が推奨されるものを使用言語別に示します。

dbinit -le オプションを使用すると、SQL Anywhere データベースで使用可能なすべての文字セット・エンコードのリストを取得できます。「初期化ユーティリティ (dbinit)」 662 ページを参照してください。

注意
次の表に示されていない言語については、UTF-8 エンコードを UCA 照合または UTF8BIN 照合と組み合わせて使用してください。

Windows プラットフォーム

言語	Windows コード・ページ	文字セット・ラベル	照合	代替照合
アラビア語	1256	Windows -1256	1256ARA	
中央および西ヨーロッパ言語	1250	Windows -1250	1250LATIN2	
デンマーク語	1252	Windows -1252	1252LATIN1	
オランダ語	1252	Windows -1252	1252LATIN1	
英語	1252	Windows -1252	1252LATIN1	
フィンランド語	1252	Windows -1252	1252SWEFIN	
フランス語	1252	Windows -1252	1252LATIN1	
ドイツ語	1252	Windows -1252	1252LATIN1	
ギリシャ語	1253	Windows -1253	1253ELL	
ヘブライ語	1253	Windows -1253	1255HEB	
イタリア語	1252	Windows -1252	1252LATIN1	
日本語	932	Windows-31J	932JPN	
韓国語	949	IBM949	949KOR	
リトアニア語	1257	Windows -1257	1257LIT	
ノルウェー語	1252	Windows -1252	1252NOR	
ポーランド語	1250	Windows -1250	1250POL	
ポルトガル語	1252	Windows -1252	1252LATIN1	
ロシア語	1251	Windows -1251	1251CYR	
中国語 (簡体文字)	936	GBK	936ZHO	

言語	Windows コード・ページ	文字セット・ラベル	照合	代替照合
スペイン語	1252	Windows -1252	1252SPA	
スウェーデン語	1252	Windows -1252	1252SWEFIN	
タイ語	874	TIS-620	874THAIBIN	
中国語 (繁体文字) - 香港	950	Big5-HKSCS	950ZHO_HK	
中国語 (繁体文字) - 台湾	950	Big5	950ZHO_TW	
トルコ語	1254	Windows -1254	1254TRK	1254TRKALT
ウクライナ語	1251	Windows -1251	1251CYR	
西ヨーロッパ言語	1252	Windows -1252	1252LATIN1	

UNIX プラットフォーム

言語	文字セット・ラベル	照合	代替照合
アラビア語	ISO_8859-6:1987	UCA	
中央および西ヨーロッパ言語	ISO_8859-2:1987	UCA	
デンマーク語	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
オランダ語	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
英語	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
フィンランド語	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
フランス語	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
ドイツ語	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
ギリシャ語	ISO_8859-7:1987	UCA	
ヘブライ語	ISO_8859-8:1988	UCA	
イタリア語	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
日本語	EUC-JP[1]	EUC_JAPAN	
韓国語	EUC-KR	EUC_KOREA	
リトアニア語	(UTF-8 を使用)	UCA または UTF8BIN	
ノルウェー語	ISO-8859-15	ISO9LATIN1	ISO1LATIN1

言語	文字セット・ラベル	照合	代替照合
ポーランド語	ISO_8859-2:1987	UCA	
ポルトガル語	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
ロシア語	ISO_8859-5:1988	UCA	
中国語 (簡体文字)	GB2312	EUC_CHINA	
スペイン語	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
スウェーデン語	ISO-8859-15	ISO9LATIN1	ISO1LATIN1
タイ語	(UTF-8 を使用)	UCA または UTF8BIN	
中国語 (繁体文字) - 香港	Big5-HKSCS	950ZHO_HK	950TWN
中国語 (繁体文字) - 台湾	EUC-TW	EUC_TAIWAN	
中国語 (繁体文字) - 台湾	Big5	950ZHO_TW	
トルコ語	ISO_8859-9:1989	920TRK	
ウクライナ語	ISO_8859-5:1988	UCA	
西ヨーロッパ言語	ISO-8859-15	ISO9LATIN1	ISO1LATIN1

[1] EUC-JP は、SQL Anywhere ラベルである Extended_UNIX_Code_Packed_Format_for_Japanese の代替ラベルです。

トルコ語文字セットと照合

トルコ語は、**I** という文字に 2 通りの表示形式があります。1 つ目の形式は、**I-dot** と呼ばれるもので、次のように表示されます。

i, İ

2 つ目の形式は、**I-no-dot** と呼ばれるもので、次のように表示されます。

ı, I

これらの文字は同じ文字の変形として表示される場合でも、トルコ語のアルファベットでは別の文字と見なされます。SQL Anywhere では、トルコ語照合 1254TRK を提供して、このような違いに対処しています。

これらの文字の大文字と小文字の変換に関するトルコ語の規則は、ANSI SQL 標準規則と互換性がありません。たとえば、トルコ語で I の小文字に相当するのは次の文字です。

```
ı
```

一方、ANSI 規格ではこれは次の文字になります。

```
i
```

このような理由から、大文字と小文字を区別しない場合に正しく一致させることができるのは、一致させるテキストがトルコ語か英語／ANSI であるかどうかによります。多くのコンテキストでは、これを区別するだけの十分な情報がないので、そのようなデータベースでは標準外の動作となることがあります。

たとえば、次の文を 1254TRK 照合を使用するデータベースに対して実行するとします。

```
SELECT * FROM syshistory   //actual table name is SYSHISTORY
SELECT * FROM fig          //actual name is FİG
```

最初の文はシステム・オブジェクトを参照しており、名前の照合には ANSI SQL 変換規則が必要です。2 番目の文はユーザ・オブジェクトを参照しており、名前の照合にはトルコ語変換規則が必要です。ここで、データベース・サーバは、オブジェクトが何であるかを確認できるまでは使用すべき変換規則を判断できず、使用する変換規則が確認できるまではオブジェクトが何であるかを判断できません。この状況においては、システムおよびユーザ・オブジェクトの両方を満足させるような解決策はありません。この例では、データベース・サーバがトルコ語照合 1254TRK を使用しており、I の小文字が I の大文字と同じであるとは見なされないので、最初の文は失敗し、2 番目の文は成功します。

トルコ語と ANSI 標準の非互換性により、トルコ語データベースのシステム・オブジェクトの参照先が*正しい*大文字と小文字のオブジェクト名、つまりオブジェクトを作成するのに使用した大文字と小文字のオブジェクト名を指定する必要があります。上記の最初の文は次のように記述します。

```
SELECT * FROM SYSHISTORY
```

実際は、文字 I のみ大文字と小文字を正しく対応させます。

別の方法として、通常のやり方ではありませんが、次のように文を記述する方法も可能です。

```
SELECT * FROM syshıstory   //I-no-dot
```

INSERT などのキーワードは、トルコ語データベースでも大文字小文字の区別がないことに注意してください。SQL Anywhere では、すべてのキーワードが英語の文字のみを使用していると認識しているので、キーワードの一致に ANSI の大文字と小文字の変換規則を使用します。また SQL Anywhere は、組み込み関数などの特定の識別子にこの方法を適用します。ただし、カタログ内に保管されている名前を持つオブジェクトは、上記のように正しい大文字と小文字または文字を使用して指定してください。

大文字と小文字を区別しないトルコ語データベースのデータ

同様の規則で、大文字小文字を区別しないトルコ語データベースのデータを管理します。たとえば、データ値が次のような場合、

```
FİG
```

上記のデータへの小文字参照は、次のようになります。

```
fig
```

同じ I-dot 文字が両方の形式で使用されます。

代替トルコ語照合 1254TRKALT

一部のアプリケーション開発者にとって、トルコ語の文字の問題が重大な問題を引き起こす場合もあります。正しい解決策は、すべてのオブジェクト参照先で大文字と小文字が正しく対応していることを確認するか、適切な文字 I を使用しているかを確認することですが、トルコ語の規則に合わせず ANSI 規則を優先した方がうまくいく場合もあります。

SQL Anywhere には、照合 1254TRKALT があります。これは、I-dot と I-no-dot を同等の文字とする以外は 1254TRK と同じです。

この変更に伴う違いをきちんと理解しておく必要があります。1254TRKALT データベースでは、次の文字列の区別がありません。

```
fig
fıg
```

これはトルコ語ユーザにとって正しくはありませんが、許容範囲として扱える場合もあります。

2 番目の問題は、ORDER BY を使用する場合に出てきます。次の文字列について考えてみます。

```
ia
ıa
ıs
is
```

1254TRK データベースでは、文字列の ORDER BY は次のような順序を生成します。

```
ıa
ıs
ia
is
```

これは、I-no-dot は I-dot より小さいからです。1254TRKALT データベースでは、次のような順序になります。

```
ia
ıa
ıs
is
```

これは、I-no-dot と I-dot が同等だからです。

第 11 章

ユーザ ID とパーミッションの管理

目次

データベースのパーミッションの概要

データベースの各ユーザは、データベースへの接続時に入力する名前 (ユーザ ID) を持っていることが必要です。ユーザ ID とパーミッションを正しく管理することによって、データベース内の情報のセキュリティやプライバシーを管理しながら作業を効率的に処理できます。

パーミッションはユーザ ID に対して与えられます。この章では、「**ユーザ**」をユーザ ID の同義語として使用します。SQL 文を使用して、データベースの新規ユーザにユーザ ID を割り当てたり、パーミッションの付与と取り消しを行ったり、ユーザの現在のパーミッションの検索を行ったりします。

個別のユーザ ID の設定

マルチユーザ・データベースにおいて、セキュリティが問題にならない場合でも、各ユーザに対して個別のユーザ ID の設定が必要な場合もあります。パーミッションは、個々のユーザだけでなく、ユーザのグループにも付与できます。グループを作成して適切なパーミッションを与えれば、管理作業にかかるオーバヘッドが非常に少なくなります。

個別のユーザ ID を使用すると、次のような利点があります。

- ログ変換ユーティリティ (dblog) によって、個々のユーザが加えた変更を必要な部分だけトランザクション・ログから抽出できます。これはトラブルシューティングのときに非常に有効です。
- Sybase Central では、どの接続がどのユーザのものであるかわかるように、有用な情報が表示されます。
- ローのロックに関するメッセージ (blocking オプションが Off の場合) の情報量が増えます。

DBA 権限の概要

データベースを作成するときには、1 つの使用可能なユーザ ID も作成されます。デフォルトでは、最初のユーザ ID は **DBA** であり、パスワードは **sql** となります (パスワードでは大文字と小文字が区別されます)。DBA ユーザの名前とパスワードを変更するには、CREATE DATABASE 文の DBA USER 句と DBA PASSWORD 句を使用するか、dbinit -dba オプションを指定します。「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』と「初期化ユーティリティ (dbinit)」 662 ページを参照してください。

ユーザ ID DBA には、データベース内で自動的に DBA 権限が設定されます。このレベルのパーミッションを持つ DBA ユーザは、データベースに関係するすべての作業を実行できます。これにはテーブルの作成、テーブル構造の変更、新規ユーザ ID の作成、ユーザからのパーミッションの取り消し、データベースのバックアップなどが含まれます。

DBA 権限を持つユーザ

DBA 権限を持つユーザは、「**データベース管理者**」になります。この章では、データベース管理者や「**DBA**」は、「*DBA 権限*」を持つ 1 人または複数のユーザを意味します。

DBA 権限を他のユーザ ID に付与または譲渡することはできますが、この章では、ユーザ ID DBA はデータベース管理者のことを指します。省略形の DBA はユーザ ID DBA と、DBA 権限を与えられた任意のユーザ ID の両方の意味に使用します。

新しいユーザの追加

DBA は、データベースに新しいユーザを追加する権限を持っています。DBA が追加したユーザには、データベース上でタスクを実行するためのパーミッションも付与されます。SQL クエリを使ってデータベース中の情報を見るだけのユーザも、データベースに情報を追加するユーザもいます。また、データベースの構造そのものを変更するユーザもいます。DBA の責任を他のユーザに分散することは多少できますが、データベース全体の管理は DBA 権限を持つ DBA の責任です。

DBA はデータベース・オブジェクトを作成して、他のユーザ ID に所有権を割り当てることができます。

注意
データに対する不正アクセスを防止するために、データベースを展開する前に DBA ユーザのパスワード (または DBA ユーザとパスワード) を変更してください。

RESOURCE 権限の概要

「**RESOURCE 権限**」とは、テーブル、ビュー、ストアド・プロシージャ、トリガなどのデータベース・オブジェクトを作成するパーミッションのことです。RESOURCE 権限をユーザに付与する権限は DBA だけが持っています。

トリガを作成するには、ユーザには RESOURCE 権限と対象テーブルの ALTER パーミッションの両方が必要です。

別の所有者のデータベース・オブジェクトを作成するためには、DBA 権限が必要です。

BACKUP 権限の概要

BACKUP 権限は、アーカイブ・バックアップやイメージ・バックアップなどを使用してデータベースやトランザクション・ログをバックアップするパーミッションです。実行するには、BACKUP 文または dbbackup ユーティリティを使用します。BACKUP 権限をユーザに付与する権限は DBA だけが持っています。「BACKUP 文」 『SQL Anywhere サーバ - SQL リファレンス』と「バックアップ・ユーティリティ (dbbackup)」 640 ページを参照してください。

VALIDATE 権限の概要

VALIDATE 権限は、データベース、テーブル、インデックス、およびチェックサムの検証を実行するパーミッションです。実行するには、VALIDATE 文または dbvalid ユーティリティを使用します。VALIDATE 権限をユーザに付与する権限は DBA だけが持っています。「VALIDATE

文」『SQL Anywhere サーバ - SQL リファレンス』と「検証ユーティリティ (dbvalid)」 756 ページを参照してください。

所有権パーミッションの概要

データベース・オブジェクトの作成者は、そのオブジェクトの所有者になります。データベース・オブジェクトの所有権は、すなわちそのオブジェクトに対してアクションを実行するパーミッションになります。これらのパーミッションは、この章に出てくる他のパーミッションのように、ユーザに割り当てられることはありません。

所有者

データベース中で新しくオブジェクトを作成したユーザは、そのオブジェクトの「**所有者**」と呼ばれます。所有者には、そのオブジェクトに対してすべての操作を行うパーミッションが自動的に与えられます。たとえば、テーブルの所有者はそのテーブルの構造を変更したり、他のユーザにテーブルのデータを更新するパーミッションを与えたりできます。

DBA は、データベース内のすべてのコンポーネントを変更できるパーミッションを持っています。したがって、他のユーザが作成したテーブルを削除することもできます。また DBA は、各データベース・オブジェクトの所有者がそのオブジェクトに関して持っているパーミッションをすべて持っています。他のユーザ用のデータベース・オブジェクトを作成することもできます。この場合、オブジェクトの所有者と CREATE 文を実行するユーザ ID が異なります。ここでは、データベース・オブジェクトの所有者と作成者を同一のユーザとして扱います。

参照

- 「パスワードのないグループ」 391 ページ

テーブルおよびビューのパーミッションの概要

テーブルおよびビューに関連した、ユーザ ID に付与されるパーミッションを次に示します。

パーミッション	説明
ALTER	テーブルの構造を変更したり、テーブルにトリガを設定したりできる。
DELETE	テーブルまたはビューからローを削除できる。
INSERT	テーブルまたはビューにローを追加できる。
REFERENCES	テーブルにインデックスを作成し、そのテーブルを参照する外部キーを作成できる。
SELECT	テーブルまたはビューの情報を見ることができる。
UPDATE	テーブルまたはビューのローを更新できる。このパーミッションは、テーブルやビューのカラムのセットに設定することもできる。

パーミッション	説明
ALL	これらのすべてのことができるパーミッション

グループ・パーミッションの概要

各ユーザに対して個別にパーミッションを設定するのは時間がかかり、またエラーの原因になります。ほとんどのデータベースでは、グループ単位のパーミッション管理の方が、個々のユーザ ID 単位の管理よりはるかに効率的です。

グループに対してパーミッションを与える方法は、個別のユーザに対する方法とまったく同じです。新しいユーザに対してそのグループのメンバシップを与えることにより、メンバシップに関連するパーミッションのセットを与えられます。

例

たとえば、会社のデータベース内で、部署ごとに (営業部やマーケティング部などの) グループを作成し、それらのグループにパーミッションを与えることができます。各営業部員は営業部グループのメンバになり、自動的にデータベースの適切な領域へアクセスできるようになります。

各ユーザ ID は複数グループのメンバとなることができ、各グループのパーミッションをすべて継承します。

個別のユーザ ID とパーミッションの管理

ここでは、新しいユーザを作成して、パーミッションを与える方法について説明します。ほとんどのデータベースでは、パーミッション管理の大半を個別のユーザに対してではなく、「**グループ**」に対して行ってください。ただし、グループは単に特別なプロパティが付加されたユーザ ID にすぎないので、グループ管理に関する説明に移る前に、この項の説明をお読みください。

新しいユーザの作成

Sybase Central と Interactive SQL のどちらでも、新しいユーザを作成できます。Sybase Central では、[ユーザとグループ] フォルダを使用してユーザやグループを管理します。Interactive SQL では、GRANT CONNECT 文を使用して新しいユーザを追加できます。どちらのツールでも、新しいユーザの作成には DBA 権限が必要です。

新しいユーザはすべて、PUBLIC グループに自動的に追加されます。新しいユーザを作成すると、次を実行できます。

- そのユーザを他のグループに追加する。「グループ・メンバシップを既存のユーザまたはグループに付与する」 388 ページを参照してください。
- テーブル、ビュー、プロシージャにそのユーザのパーミッションを設定する。「個別のユーザ ID とパーミッションの管理」 374 ページを参照してください。
- そのユーザをパブリッシャ、またはデータベースのリモート・ユーザとして設定する。「SQL Remote パーミッションの管理」 『SQL Remote』を参照してください。

新しいユーザに対する初期パーミッション

デフォルトで新しいユーザに割り当てられるパーミッションには以下が含まれます。

- データベースへの接続 (ユーザのパスワードが指定されていることが前提)
- システム・ビューに格納されているデータの表示
- 大部分のシステム・ストアド・プロシージャの実行

データベースにあるテーブルにアクセスするには、新しいユーザにパーミッションを付与する必要があります。

DBA は、特殊な PUBLIC ユーザ・グループに対してパーミッションを割り当てることで、新しいユーザに自動的に与えられるパーミッションを設定できます。「特殊なグループ」 392 ページを参照してください。

ユーザ ID とパスワードに関する制限事項

ユーザ ID とパスワードを作成するとき、名前やパスワードに対して次の制限が適用されます。

- 空白スペース、一重引用符、または二重引用符で始まる名前
- 空白スペースで終わる名前
- セミコロンを含む名前

♦ **新しいユーザを作成するには、次の手順に従います (Sybase Central の場合)。**

1. [ユーザとグループ] フォルダを開きます。
2. [ファイル] メニューから、[新規] – [ユーザ] を選択します。

   [ユーザの作成] ウィザードが表示されます。
3. ウィザードの指示に従います。

♦ **新しいユーザを作成するには、次の手順に従います (SQL の場合)。**

1. DBA 権限を持つユーザとしてデータベースに接続します。
2. GRANT CONNECT TO 文を実行します。

例

ユーザ ID M_Haneef とパスワード Welcome を使用して、データベースに新しいユーザを追加します。

```
GRANT CONNECT TO M_Haneef
IDENTIFIED BY Welcome;
```

参照

♦ 「GRANT 文」 『SQL Anywhere サーバ - SQL リファレンス』

パスワードの設定

ユーザは、データベースに接続可能なパスワードを持っていなければなりません。パスワードの長さは 255 バイト以内で、セミコロンや前後のスペースを含めることはできません。パスワードでは大文字と小文字が区別されます。

作成または変更したパスワードは、UTF-8 に変換されてからハッシュされ、データベースに保存されます。データベースをアンロードし、別の文字セットを使用するデータベースに再ロードした場合でも、既存のパスワードは機能します。パスワードには 7 ビット ASCII 文字を使用することをおすすめします。それ以外の文字を使用すると、サーバがクライアントの文字セットを UTF-8 に変換できない場合、パスワードが機能しないことがあります。

パスワードの変更

GRANT 文を使用して自分のパスワードを変更できます。DBA 権限を持っていれば、他のユーザのパスワードも変更できます。たとえば、次に示すコマンドは、ユーザ ID M_Haneef のパスワードを new_password に変更します。

```
GRANT CONNECT TO M_Haneef
IDENTIFIED BY new-password;
```


DBA パスワードの変更

ユーザ ID DBA のデフォルトのパスワードは、すべてのデータベースで sql です。データベースに対する不正なアクセスを防ぐために、このパスワードを変更してください。次のコマンドは、DBA ユーザのパスワードを new_password に変更します。

```
GRANT CONNECT TO DBA
IDENTIFIED BY new_password;
```

ユーザとグループのオプションの設定

Sybase Central では、ユーザやグループのための設定オプションは、[ユーザのオプション] ダイアログと [グループのオプション] ダイアログ (データベースのオプションを設定するためのダイアログと同じもの) にあります。Interactive SQL では、SET OPTION 文でオプションを指定できます。

♦ ユーザまたはグループのオプションを設定するには、次の手順に従います (Sybase Central の場合)。

1. データベースに接続し、[ユーザとグループ] フォルダを開きます。
2. 対象のユーザまたはグループを選択し、[ファイル] – [オプション] を選択します。
3. 値を編集します。
4. [恒久的な設定を行う] をクリックします。

♦ ユーザまたはグループのオプションを設定するには、次の手順に従います (SQL の場合)。

- SET OPTION 文で対象のプロパティを指定します。

参照

- 「データベース・オブジェクトのプロパティの設定」『SQL Anywhere サーバ - SQL の使用法』
- 「データベース・オプション」 405 ページ

DBA および RESOURCE 権限の付与

DBA と RESOURCE の権限は同じ方法で付与できます。

♦ ユーザ ID に RESOURCE 権限を付与するには、次の手順に従います。

1. DBA としてデータベースに接続します。
2. 次の SQL 文を入力して実行します。

```
GRANT RESOURCE TO user-id;
```

DBA 権限の場合は、次の SQL 文を使います。

GRANT DBA TO *user-id*;

注意

- DBA または RESOURCE の権限をデータベース・ユーザに付与できるのは DBA だけです。
- DBA 権限は非常に強力です。この権限を持つユーザは、データベース中のすべての情報にアクセスできるだけでなく、データベースに対するすべてのアクションを実行できます。DBA 権限を付与するユーザは、少数に限定してください。
- DBA 権限を必要とするユーザには、ユーザ ID を 2 つ割り当てることを検討してください。この場合、一方だけに DBA 権限を付与し、必要なときだけ DBA として接続するようにします。
- RESOURCE 権限はユーザに、テーブル、ビュー、インデックス、プロシージャ、トリガなどの新規データベース・オブジェクトの作成を許可します。

テーブルに対するパーミッションの付与

個々のテーブルにパーミッションのセットを割り当てて、これらのパーミッションの組み合わせをユーザに付与すると、テーブルへのアクセスを定義できます。

Sybase Central または Interactive SQL のいずれかを使用して、パーミッションを設定できます。Interactive SQL では、GRANT 文を使用してテーブルに以下のパーミッションを付与できます。

- ALTER パーミッションによって、ユーザはテーブルの構造を変更したりテーブル上でトリガを作成したりできます。REFERENCES パーミッションでは、テーブル上にインデックスを作成し、さらに外部キーを作成できます。これらのパーミッションにより、データベース・スキーマの変更権限が付与されます。したがって、ほとんどのユーザは付与の対象になりません。また、これらのパーミッションはビューには適用されません。
- DELETE、INSERT、UPDATE の各パーミッションは、テーブルのデータを修正する権限を付与します。
- SELECT パーミッションは、テーブルのデータを見る権限を付与しますが、変更するためのパーミッションは付与しません。
- ALL パーミッションは、これらすべてのパーミッションを付与します。
- REFERENCES、SELECT、および UPDATE パーミッションは、対象をテーブルまたはビューのカラムのセットに限定できます。

♦ テーブルまたはカラムに対するパーミッションを付与するには、次の手順に従います (Sybase Central の場合)。

1. データベースに接続します。
2. データベースの [テーブル] フォルダを開きます。
3. テーブルを選択して、[ファイル] – [プロパティ] を選択します。

4. [テーブル] プロパティ・シートの [パーミッション] タブで、テーブルに対するパーミッションを設定します。
   - [付与] をクリックし、パーミッションを付与するユーザまたはグループを選択します。
   - ユーザまたはグループの横にあるフィールドをクリックして、特定のパーミッションを設定します。パーミッションはチェック・マークによって示されます。また、付与されるオプションは、2 つのプラス記号 (+) が付いたチェック・マークによって示されます。
   - ユーザを選択して [参照]、[選択]、[更新] の横にある [変更] ボタンをクリックし、個々のカラムに対するパーミッションのタイプを設定します。
   - リストからユーザまたはグループを選択し、[取り消し] をクリックしてパーミッションをすべて取り消します。

ヒント
[ユーザ] または [グループ] のプロパティ・シートから、パーミッションを割り当てることもできます。一度に複数のユーザやグループにパーミッションを割り当てるには、テーブルのプロパティ・シートを使用します。一度に複数のテーブルにパーミッションを割り当てるには、ユーザのプロパティ・シートを使用します。

♦ テーブルまたはカラムに対するパーミッションを付与するには、次の手順に従います (SQL の場合)。

1. DBA またはテーブルの所有者としてデータベースに接続します。
2. GRANT 文を実行してパーミッションを割り当てます。

   「GRANT 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

例 1

テーブル・パーミッションはすべて、よく似た方法で付与されます。たとえば、次の手順では、M_Haneef に sample_table というテーブルからローを削除するパーミッションを与えることができます。

1. DBA または sample_table の所有者としてデータベースに接続します。
2. 次の SQL 文を実行します。

```
GRANT DELETE
ON sample_table
TO M_Haneef;
```

例 2

次の手順で、sample_table という名前のテーブルに含まれるカラム column_1 と column_2 だけを更新するパーミッションを M_Haneef に与えることができます。

1. DBA または sample_table の所有者としてデータベースに接続します。
2. 次の SQL 文を実行します。

```
GRANT UPDATE (column_1, column_2)
ON sample_table
TO M_Haneef;
```

テーブル・パーミッションの対象は、テーブル内の全データに制限されます。ただし、REFERENCES、SELECT、および UPDATE パーミッションはカラムのサブセットに付与できます。さらに細かいユーザ・パーミッションを設定するには、テーブルに対してアクションを実行するプロシージャを作成し、そのプロシージャを実行するパーミッションをユーザに与えます。

参照

- 「GRANT 文」『SQL Anywhere サーバ - SQL リファレンス』

ビューに対するパーミッションの付与

ビューでのパーミッションの設定は、テーブルで設定する場合と似ています。

関連する SQL 文の詳細については、「テーブルに対するパーミッションの付与」 377 ページを参照してください。

次の条件の 1 つまたは複数を満たす場合に、ユーザはビューを介して操作を実行できます。

- ビューに対する特定の操作のパーミッションが、DBA によってユーザに正しく付与されている場合。
- すべてのベース・テーブルに対する特定の操作のパーミッションを、ユーザが正しく保持する場合。
- ビューに対する特定の操作のパーミッションが、DBA 以外のユーザによって正しく付与された場合。このユーザは、ビューの所有者であるか、またはビューに対する適切なパーミッション WITH GRANT OPTION を所有する必要があります。ビューの所有者は次のいずれかです。
  - DBA
  - DBA 以外の、ビューによって参照されるすべてのベース・テーブルの所有者
  - DBA 以外であり、ビューによって参照されるいくつか、またはすべてのベース・テーブルの所有者ではない者。ただし、ビューの所有者は、所有していないベース・テーブルに WITH GRANT OPTION で SELECT パーミッションを持ち、所有していないベース・テーブルに WITH GRANT OPTION で操作に対するその他の必要なパーミッションを持っています。

    所有者がベース・テーブルに対するパーミッション (WITH GRANT OPTION) を保持する代わりに、PUBLIC にパーミッションが付与される場合もあります。これには、システム・テーブルに対する SELECT パーミッションが含まれます。

UPDATE パーミッションは、ビュー全体とビュー内の個々のカラムのどちらにも付与できます。

♦ ビューに対するパーミッションを付与するには、次の手順に従います (Sybase Central の場合)。

1. データベースに接続します。
2. データベースの [ビュー] フォルダを開きます。
3. ビューを選択して、[ファイル] – [プロパティ] を選択します。
4. [ビュー] プロパティ・シートの [パーミッション] タブで、ビューに対するパーミッションを設定します。
   - [付与] をクリックし、全パーミッションを付与するユーザまたはグループを選択します。
   - ユーザまたはグループの横にあるフィールドをクリックして、特定のパーミッションを設定します。パーミッションはチェック・マークによって示されます。また、付与されるオプションは、2 つのプラス記号 (+) が付いたチェック・マークによって示されます。
   - リストからユーザまたはグループを選択し、[取り消し] をクリックしてパーミッションをすべて取り消します。

ヒント
[ユーザ] または [グループ] のプロパティ・シートから、パーミッションを割り当てることもできます。一度に複数のユーザやグループにパーミッションを割り当てるには、ビューのプロパティ・シートを使用します。一度に複数のビューにパーミッションを割り当てるには、ユーザまたはグループのプロパティ・シートを使用します。

参照

- 「GRANT 文」 『SQL Anywhere サーバ - SQL リファレンス』

パーミッションを付与する権利をユーザに付与する

テーブルとビューに対するパーミッションは、WITH GRANT OPTION を付けて割り当てられます。このオプションは、パーミッションを他のユーザに引き渡す権利を与えます。

Sybase Central では、パーミッション付与についてのオプションを指定できます。ユーザ、グループ、テーブルのプロパティ・シートで [パーミッション] タブをクリックし、表示されたフィールドをダブルクリックして 2 つの + 記号が付いたチェック・マークを表示します。

注意
WITH GRANT OPTION はユーザにのみ指定できます。グループのメンバは、グループに付与されている WITH GRANT OPTION を継承しません。

例

M_Haneef にテーブル sample_table からローを削除するパーミッションを付与し、さらにこのパーミッションを他のユーザに渡す権利を付与するには、次の手順に従います。

1. DBA または sample_table の所有者としてデータベースに接続します。
2. 次の SQL 文を入力して実行します。

```
GRANT DELETE ON sample_table
TO M_Haneef
WITH GRANT OPTION;
```

「GRANT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

参照

- 「テーブルに対するパーミッションの付与」 377 ページ
- 「グループのパーミッション」 390 ページ

プロシージャに対するパーミッションの付与

DBA またはプロシージャの所有者は、ストアド・プロシージャを実行するパーミッションを付与できます。EXECUTE パーミッションは、プロシージャに対して付与できる唯一のパーミッションです。

プロシージャを実行するパーミッションを与える方法は、テーブルとビューにパーミッションを与える方法と似ています。ただし、GRANT 文の WITH GRANT OPTION 句は、プロシージャに対するパーミッションの付与に適用されません。

Sybase Central または Interactive SQL のいずれかを使用して、パーミッションを設定できます。

♦ プロシージャに対するパーミッションを付与するには、次の手順に従います (Sybase Central の場合)。

1. データベースに接続します。
2. データベースの [プロシージャとファンクション] フォルダを開きます。
3. プロシージャを選択して、[ファイル] – [プロパティ] を選択します。
4. [プロシージャ] プロパティ・シートの [パーミッション] タブで、プロシージャに対するパーミッションを設定します。
   - [付与] をクリックし、プロシージャに対するすべてのパーミッションを付与するユーザまたはグループを選択します。
   - ユーザのとなりの [実行] カラムをクリックし、パーミッションを付与するかどうかを切り替えます。
   - リストからユーザまたはグループを選択し、[取り消し] をクリックしてパーミッションをすべて取り消します。

ヒント
[ユーザ] または [グループ] のプロパティ・シートから、パーミッションを割り当てることもできます。一度に複数のユーザやグループにパーミッションを割り当てるには、プロシージャのプロパティ・シートを使用します。一度に複数のプロシージャにパーミッションを割り当てるには、ユーザまたはグループのプロパティ・シートを使用します。

♦ **プロシージャに対するパーミッションを付与するには、次の手順に従います (SQL の場合)。**

1. DBA またはプロシージャの所有者としてデータベースに接続します。
2. GRANT EXECUTE ON 文を実行します。

例

次の手順では、プロシージャ my_procedure を実行するパーミッションを M_Haneef に与えることができます。

1. DBA または my_procedure プロシージャの所有者としてデータベースに接続します。
2. 次の SQL 文を入力して実行します。

```
GRANT EXECUTE
ON my_procedure
TO M_Haneef;
```

プロシージャの EXECUTE パーミッション

プロシージャは所有者のパーミッションのもとで実行されます。テーブルの情報を更新するプロシージャが正しく実行されるのは、そのプロシージャの所有者が対象となるテーブルの UPDATE パーミッションを持っている場合のみです。

プロシージャの所有者が適切なパーミッションを持っていれば、そのプロシージャの EXECUTE パーミッションを割り当てられているユーザは、基本となるテーブルに対するパーミッションの有無にかかわらず、そのプロシージャを実行できます。プロシージャを使用すると、テーブルに対する一般的なパーミッションがなくても、ユーザがテーブルに一定の作業を行う許可を与えることができます。

参照

- 「GRANT 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「テーブルに対するパーミッションの付与」 377 ページ

トリガの EXECUTE パーミッション

ユーザのアクションに応じて、サーバがトリガを実行します。トリガの実行にパーミッションは必要ありません。トリガの実行は、関連するテーブルの作成者のパーミッションで行われます。

「トリガを実行するためのパーミッション」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

REMOTE パーミッションの付与と取り消し

Sybase Central では、ユーザとグループの REMOTE パーミッションを管理できます。REMOTE パーミッションを使用すると、通常のユーザとグループを SQL Remote レプリケーション設定においてリモート・ユーザにすることができます。これにより、パブリッシュするデータベースとレプリケーション・メッセージを交換できます。

REMOTE パーミッションの付与

REMOTE パーミッションは、データベースにメッセージ・タイプを少なくとも 1 つは定義しないかぎり、ユーザまたはグループに付与することはできません。

REMOTE パーミッションをグループに付与するには、グループ内のすべてのユーザに REMOTE パーミッションを明示的に与える必要があります。グループのメンバは、REMOTE パーミッションを継承しません。

REMOTE パーミッションの取り消し

REMOTE パーミッションを取り消すと、リモート・ユーザは通常のユーザに戻ります。REMOTE パーミッションを取り消すと、そのユーザのサブスクリプションがすべてのパブリケーションから自動的に削除されます。

♦ ユーザに REMOTE パーミッションを付与するには、次の手順に従います (Sybase Central の場合)。

1. データベースに接続します。
2. [ユーザとグループ] フォルダを開きます。
3. 対象のユーザを選択し、[ファイル] – [リモート・ユーザに変更] を選択します。

   [リモート・ユーザに変更] ダイアログが表示されます。
4. 表示されるダイアログで、必要な値を入力します。

ユーザに REMOTE パーミッションを付与すると、パブリケーションにそのユーザのサブスクリプションを作成できます。

♦ リモート・ユーザから REMOTE パーミッションを取り消すには、次の手順に従います。

1. [ユーザとグループ] フォルダまたは [SQL Remote ユーザ] フォルダを開きます。
2. 対象のリモート・ユーザを選択し、[ファイル] – [リモートの取り消し] を選択します。

「SQLRemote の概念」 『SQL Remote』を参照してください。

ユーザ・パーミッションの取り消し

ユーザのパーミッションは、実際には与えられたパーミッションと取り消されたパーミッションの組み合わせです。パーミッションの付与と取り消しを使って、データベースのユーザ・パーミッションを管理できます。

REVOKE 文は、GRANT 文とはまったく逆の処理を行います。次は、M_Haneef が my_procedure を実行できなくなるコマンドの例です。

```
REVOKE EXECUTE
ON my_procedure
FROM M_Haneef;
```

DBA またはプロシージャの所有者が、このコマンドを発行してください。

グループにユーザを追加すると、ユーザはそのグループに割り当てられたすべてのパーミッションを継承します。SQL Anywhere では、ユーザがグループのメンバとして継承するパーミッションのサブセットを取り消すことはできません。取り消すことができるのは、GRANT 文によって明示的に付与されるパーミッションだけです。異なるユーザに別々のパーミッションを付与する必要がある場合、適切なパーミッションを持つグループを別々に作成するか、必要なパーミッションを各ユーザに明示的に付与できます。

接続パーミッションまたは別のユーザからのテーブル・パーミッションを取り消す場合、他のユーザはそのデータベースに接続できません。dbo からの接続パーミッションの取り消しはできません。

sample_table からローを削除するパーミッションは、次のコマンドを発行することで取り消すことができます。

```
REVOKE DELETE
ON sample_table
FROM M_Haneef;
```

参照

- 「テーブルに対するパーミッションの付与」 377 ページ
- 「ビューに対するパーミッションの付与」 379 ページ
- 「プロシージャに対するパーミッションの付与」 381 ページ

データベースからユーザを削除する

Sybase Central と Interactive SQL のどちらでも、データベースからユーザを削除できます。ユーザの削除中は、対象ユーザはデータベースに接続できません。

ユーザを削除すると、そのユーザが所有するテーブルなどのデータベース・オブジェクトもすべて削除されます。

ユーザを削除できるのは DBA だけです。

♦ データベースからユーザを削除するには、次の手順に従います (Sybase Central の場合)。

1. [ユーザとグループ] フォルダを開きます。
2. 対象のユーザを選択し、[編集] – [削除] を選択します。

♦ データベースからユーザを削除するには、次の手順に従います (SQL の場合)。

- REVOKE CONNECT FROM 文を実行します。

例

データベースからユーザ M_Haneef を削除します。

```
REVOKE CONNECT FROM M_Haneef;
```

参照

- 「REVOKE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「ユーザ・パーミッションの取り消し」 383 ページ
- 「データベースからグループを削除する」 393 ページ

接続されたユーザの管理

Sybase Central で作業している場合は、データベースに接続されたすべてのユーザを追跡できます。接続されたユーザのプロパティを表示したり、必要に応じてその接続を切断したりできます。

♦ データベースに接続されたすべてのユーザをリストするには、次の手順に従います。

- 左ウィンドウ枠のデータベースを選択し、右ウィンドウ枠の [接続されたユーザ] タブをクリックします。

  このタブは、接続に使用しているアプリケーション (Sybase Central、Interactive SQL、カスタム・クライアント・アプリケーションなど) に関係なく、指定したデータベースに現在接続しているすべてのユーザを表示します。

♦ ユーザとデータベースの接続に関するプロパティを検査するには、次の手順に従います。

1. 左ウィンドウ枠のデータベースを選択し、右ウィンドウ枠の [接続されたユーザ] タブをクリックします。
2. 対象のユーザを選択し、[ファイル] – [プロパティ] を選択します。
3. 対象のプロパティを検査します。

♦ データベースとユーザの接続を切断するには、次の手順に従います。

1. 左ウィンドウ枠のデータベースを選択し、右ウィンドウ枠の [接続されたユーザ] タブをクリックします。
2. 対象のユーザを選択し、[ファイル] – [切断] を選択します。

グループの管理

前述の項で説明した個別ユーザのパーミッション管理を理解すれば、グループの管理は簡単です。グループは、個々のユーザとまったく同じようにユーザ ID で識別されます。ただし、グループ・ユーザ ID には、メンバを持つことを許可するパーミッションがあります。

権限の継承

グループにパーミッションを付与したり、テーブル、ビュー、プロシージャに対するパーミッションをグループから取り消したりすると、グループのメンバ全員がその変更を継承します。次の権限は継承されません。これらの権限は、必要に応じて各ユーザ ID に個別に割り当てる必要があります。

- DBA
- BACKUP
- VALIDATE
- GROUP
- RESOURCE
- REMOTE
- REMOTE DBA
- CONSOLIDATE
- PUBLISH

注意
WITH GRANT OPTION はユーザにのみ指定できます。グループのメンバは、グループに付与されている WITH GRANT OPTION を継承しません。

グループは、単に特別なパーミッションを持つユーザ ID にすぎません。グループに対するパーミッションの付与と取り消しは、通常のユーザとまったく同じ方法で実行します。「個別のユーザ ID とパーミッションの管理」 374 ページを参照してください。

グループの階層を構成し、各グループが親グループからパーミッションを継承するようにできます。つまり、グループは他のグループのメンバになることもできます。また、各ユーザ ID は複数のグループに属することができます。したがって、ユーザとグループの関係は多対多になります。

パスワードなしのグループも作成できます。これによって、グループ・ユーザ ID を使用したデータベースへの接続を防ぐことができます。

セキュリティ機能の詳細については、「パスワードのないグループ」 391 ページを参照してください。

データベース・オブジェクトのプロパティの変更については、「データベース・オブジェクトのプロパティの設定」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

グループへの REMOTE パーミッションの付与については、「REMOTE パーミッションの付与と取り消し」 383 ページを参照してください。

グループの作成

Sybase Central および Interactive SQL のどちらでも、新しいグループを作成できます。新しいグループの作成には DBA 権限が必要です。

♦ **新しいグループを作成するには、次の手順に従います (Sybase Central の場合)。**

1. [ユーザとグループ] フォルダをクリックします。
2. [ファイル] メニューから、[新規] – [グループ] を選択します。

   [グループ作成] ウィザードが表示されます。
3. ウィザードの指示に従います。

♦ **新しいグループを作成するには、次の手順に従います (SQL の場合)。**

- GRANT GROUP TO 文を実行します。この文に指定したユーザ ID が作成されていない場合、文は失敗します。

例

ユーザ ID personnel を作成します。

```
GRANT CONNECT
TO personnel
IDENTIFIED BY group_password;
```

ユーザ ID personnel をグループにします。

```
GRANT GROUP TO personnel;
```

参照

- 「GRANT 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「新しいユーザの作成」 374 ページ

グループ・メンバシップを既存のユーザまたはグループに付与する

Sybase Central および Interactive SQL のどちらでも、既存のユーザをグループに追加したり、グループを別のグループに追加したりできます。Sybase Central では、ユーザまたはグループの右ウィンドウ枠でグループ・メンバシップを制御できます。Interactive SQL では、GRANT 文を使用してユーザをグループのメンバにできます。

グループのメンバシップを与えられたユーザは、そのグループに関連したテーブル、ビュー、プロシージャに対するパーミッションをすべて引き継ぎます。

グループのメンバシップを付与できるのは DBA だけです。

♦ ユーザまたはグループを別のグループに追加するには、次の手順に従います (Sybase Central の場合)。

1. [ユーザとグループ] フォルダを開きます。
2. 別のグループに追加するユーザまたはグループを選択し、[ファイル] メニューから [新規] – [メンバシップ] を選択します。

   [新しいメンバシップ] ダイアログが表示されます。
3. 対象のグループを選択して、[OK] をクリックします。
4. 選択したユーザまたはグループの右ウィンドウ枠の [メンバシップ] タブに、グループのメンバシップが表示されます。

♦ ユーザまたはグループを別のグループに追加するには、次の手順に従います (SQL の場合)。

- 対象のグループとユーザを指定した GRANT MEMBERSHIP IN GROUP 文を実行します。

例

ユーザ M_Haneef に、グループ personnel のメンバシップを付与します。

```
GRANT MEMBERSHIP
IN GROUP personnel
TO M_Haneef;
```

参照

- 「GRANT 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「新しいユーザの作成」 374 ページ

グループ・メンバシップの取り消し

Sybase Central および Interactive SQL のどちらでも、グループからユーザまたはグループを取り除くことができます。

グループからユーザまたはグループを取り除いても、データベース (または他のグループ) からは削除されません。データベースから削除するには、ユーザまたはグループ自体を削除する必要があります。

グループのメンバシップを取り消せるのは DBA だけです。

グループにユーザを追加すると、ユーザはそのグループに割り当てられたすべてのパーミッションを継承します。SQL Anywhere では、ユーザがグループのメンバとして継承するパーミッションのサブセットを取り消すことはできません。取り消すことができるのは、GRANT 文によって明示的に付与されるパーミッションだけです。異なるユーザに別々のパーミッションを付与する必要がある場合、適切なパーミッションを持つグループを別々に作成するか、必要なパーミッションを各ユーザに明示的に付与できます。

♦ ユーザまたはグループを別のグループから取り除くには、次の手順に従います (Sybase Central の場合)。

1. [ユーザとグループ] フォルダを開きます。
2. 対象のユーザまたはグループを選択し、右ウィンドウ枠の [メンバシップ] タブをクリックします。
3. 対象のグループを選択し、[ファイル] – [メンバシップの削除] を選択します。

   そのユーザまたはグループが、このグループから削除されます。

ヒント
上記の操作は、グループを選択して、右ウィンドウ枠の [メンバシップ] タブをクリックし、削除したいユーザまたはグループを選択して、ポップアップ・メニューから [メンバシップの削除] を選択することで実行できます。

♦ ユーザまたはグループを別のグループから取り除くには、次の手順に従います (SQL の場合)。

· 対象のグループとユーザを指定した REVOKE MEMBERSHIP IN GROUP 文を実行します。

例

グループ personnel からユーザ M_Haneef を取り除きます。

```
REVOKE MEMBERSHIP
IN GROUP personnel
FROM M_Haneef;
```

参照

- 「REVOKE 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「新しいユーザの作成」 374 ページ
- 「データベースからユーザを削除する」 384 ページ
- 「データベースからグループを削除する」 393 ページ

グループのパーミッション

グループへのパーミッションは、通常のユーザ ID とまったく同じ方法で与えることができます。テーブル、ビュー、プロシージャに対するパーミッションは、他のグループとそのメンバも含めて、グループのメンバに継承されます。グループのパーミッションには、データベース管理者が気を付けなければならない点がいくつかあります。

注意

グループのメンバは、DBA、RESOURCE、GROUP の各パーミッションを継承しません。ユーザ ID personnel が RESOURCE 権限を持っている場合でも、personnel のメンバは RESOURCE 権限を持ちません。

データベース・オブジェクトの所有者は単一のユーザ ID に属し、グループ・メンバには受け継がれません。ユーザ ID personnel がテーブルを作成した場合は、ユーザ ID personnel がそのテー

ブルの所有者となり、テーブルに変更を加える権限だけでなく、テーブルに関する権限を他のユーザに与える権限も保持します。personnel のメンバである他のユーザ ID は、このテーブルの所有者ではなく、これらの権利を持ちません。付与されたパーミッションのみを継承します。たとえば、DBA またはユーザ ID personnel が SELECT パーミッションをユーザ ID personnel に明示的に付与した場合は、すべてのグループ・メンバが、そのテーブルに対する SELECT パーミッションを継承します。

注意
WITH GRANT OPTION はユーザにのみ指定できます。グループのメンバは、グループに付与されている WITH GRANT OPTION を継承しません。

グループが所有するテーブルの参照

データベース中のテーブルとプロシージャを検索するのにグループを使うことができます。すべてのユーザがグループ PUBLIC に属し、グループ PUBLIC が SYSGROUPS ビューを所有する SYS グループに属するため、次に示すクエリは常にビュー SYS.SYSGROUPS を検索します。

```
SELECT * FROM SYSGROUPS;
```

SYSGROUPS ビューには、データベース中のグループ・メンバシップを示す *group_name* と *member_name* のペアのリストが入っています。

テーブル employees がユーザ ID personnel によって所有されており、**M_Haneef** が personnel グループのメンバである場合は、**M_Haneef** はテーブル employees を SQL 文で単に employees として参照できます。personnel グループのメンバでないユーザは、「**修飾された**」名前 personnel.employees を使用する必要があります。

テーブルを所有するグループの作成

名前を修飾しないでテーブルにアクセスできるように、テーブルを所有することだけが目的のグループを作成することをおすすめします。このグループには何のパーミッションも与えませんが、すべてのユーザをこのグループのメンバにします。次にパーミッション・グループを作成し、ユーザを適宜それらのパーミッション・グループのメンバにします。

ユーザがグループ所有のテーブルと同じ名前のテーブルを所有している場合、修飾されていないテーブル名は、グループが所有するテーブルではなくユーザが所有するテーブルを表します。同様に、ユーザが同じ名前のテーブルを所有する複数のグループに属している場合、そのユーザはテーブル名を修飾する必要があります。

「データベース・オブジェクトの名前とプレフィクス」 394 ページを参照してください。

パスワードのないグループ

グループのユーザ ID に属すユーザには、何らかのパーミッションがあります。グループに属しているユーザは、グループのユーザ ID で作成されたデータベース内のすべてのテーブルに対して、所有者のパーミッションを持ちます。

他のユーザ ID がグループ・メンバシップを変更できるようにするのではなく、DBA だけがグループとグループのデータベース・オブジェクトを処理するようにデータベースを設定できます。この場合は、グループの作成時にそのグループのユーザ ID で接続できないよう指定します。そのためには、次のように GRANT CONNECT 文をパスワードなしで入力します。次の文は、ユーザ ID personnel を作成します。

```
GRANT CONNECT
TO personnel;
```

このユーザ ID にグループ・パーミッションを付与し、他のユーザ ID にグループのメンバシップを付与して personnel に与えられているパーミッションを継承させることができます。ただし、ユーザ ID personnel には有効なパスワードがないので、このユーザ ID を使用してデータベースに接続することはできません。

ユーザがユーザ ID personnel を使用してデータベースに接続できない場合でも、このユーザ ID はデータベース・オブジェクトの所有者になることができます。CREATE TABLE 文、CREATE PROCEDURE 文、CREATE VIEW 文では、文を実行する以外のユーザとしてオブジェクトの所有者を指定できます。この所有権の割り当てを実行できるのは、DBA だけです。

特殊なグループ

データベースを作成すると、SYS、PUBLIC、dbo の各グループも自動的に作成されます。どのグループもパスワードを持たないため、SYS、PUBLIC、dbo でデータベースに接続することはできません。しかし、これらのグループはデータベース中で重要な働きをします。

SYS グループ

SYS グループは、データベース・オブジェクトとユーザ ID のすべてを含む、データベース構造を記述するシステム・テーブルとビューを所有します。

システム・テーブルとビューの説明およびテーブルへのアクセス権については、「テーブル」『SQL Anywhere サーバ - SQL リファレンス』と「Sybase Central のシステム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

PUBLIC グループ

PUBLIC グループは、システム・テーブルに対する SELECT パーミッションを持ちます。また、PUBLIC グループは SYS グループのメンバであり、システム・テーブルとビューへの読み込みアクセス権を受け継いでいます。したがって、データベースのユーザは誰でもデータベース・スキーマについて知ることができます。このアクセスを制限するには、PUBLIC から SYS グループのメンバシップを取り消します。

新しいユーザ ID は自動的に PUBLIC グループのメンバとなり、グループのパーミッション、特に DBA によりグループに与えられたパーミッションをすべて受け継ぎます。必要に応じてユーザごとに PUBLIC グループのメンバシップを取り消すこともできます。

dbo グループ

dbo グループは、多数のシステム・ストアド・プロシージャやビューを所有しています。dbo グループは、SYS グループのメンバです。PUBLIC グループは、dbo グループのメンバです。dbo グループは、Ultra Light と Mobile Link に使用するテーブルも所有しています。

データベースからグループを削除する

Sybase Central と Interactive SQL のどちらでも、データベースからグループを削除できます。

データベースからのユーザまたはグループの削除は、それらを別のグループから*取り除く*ことは異なります。データベースからグループを削除しても、データベースからそのグループのメンバは削除*されません*。ただし、削除されたグループに対するメンバシップはなくなります。

グループを削除できるのは DBA だけです。

- **データベースからグループを削除するには、次の手順に従います (Sybase Central の場合)。**

1. [ユーザとグループ] フォルダを開きます。
2. 対象のグループを選択し、[編集] – [削除] を選択します。

- **データベースからグループを削除するには、次の手順に従います (SQL の場合)。**

1. データベースに接続します。
2. REVOKE CONNECT FROM 文を実行します。

例

データベースからグループ personnel を削除します。

```
REVOKE CONNECT FROM personnel;
```

参照

- 「REVOKE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「ユーザ・パーミッションの取り消し」 383 ページ
- 「データベースからユーザを削除する」 384 ページ

データベース・オブジェクトの名前とプレフィクス

データベース・オブジェクトの名前は識別子である必要があります。

有効な識別子の規則については、「識別子」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

このマニュアルでは、クエリと SQL 文のサンプルを通じて、データベース・オブジェクトの簡略名を使用します。次に例を示します。

```
SELECT *
FROM Employees;
```

テーブル、プロシージャ、ビューにはすべて所有者があります。ユーザ ID DBA は、サンプル・データベース内のテーブルを所有します。場合によっては、オブジェクト名に所有者のユーザ ID をプレフィクスとして付ける必要があります。次に例を示します。

```
SELECT *
FROM DBA.Employees;
```

この Employees テーブルの参照を「**修飾された**」状態であるといいます。単にオブジェクト名を示すだけでよい場合もあります。この項では、どのような場合にテーブル、ビュー、プロシージャに所有者名をプレフィクスとして付ける必要があるかを説明します。

データベース・オブジェクトを参照するときプレフィクスが必要ないのは、次の場合です。

- 自分がデータベース・オブジェクトの所有者である場合。
- 自分がデータベース・オブジェクトを所有するグループのメンバである場合。

例

企業のデータベースを例に説明します。ユーザ ID company がすべてのテーブルを作成し、このユーザ ID はデータベース管理者に属するので、DBA 権限を持ちます。

```
GRANT CONNECT TO Company
IDENTIFIED BY secret;
GRANT DBA TO Company;
```

ユーザ ID company が、データベース内のテーブルを次のように作成しました。

```
CONNECT USER company IDENTIFIED BY secret;
CREATE TABLE company.Customers ( ... );
CREATE TABLE company.Products ( ... );
CREATE TABLE company.Orders ( ... );
CREATE TABLE company.Invoices ( ... );
CREATE TABLE company.Employees ( ... );
CREATE TABLE company.Salaries ( ... );
```

会社の全員がすべての情報にアクセスできるようにはしません。営業部の 2 人のユーザ ID Joe と Sally に、テーブル Customers、Products、Orders へのアクセス権限を与える場合を考えてみます。これを行うには、Sales グループを作成します。

```
GRANT CONNECT TO Sally IDENTIFIED BY xxxxx;
GRANT CONNECT TO Joe IDENTIFIED BY xxxxx;
GRANT CONNECT TO Sales IDENTIFIED BY xxxxx;
```

```
GRANT GROUP TO Sales;
GRANT ALL ON Customers TO Sales;
GRANT ALL ON Orders TO Sales;
GRANT SELECT ON Products TO Sales;
GRANT MEMBERSHIP IN GROUP Sales TO Sally;
GRANT MEMBERSHIP IN GROUP Sales TO Joe;
```

これで Joe と Sally はこれらのテーブルを使用するパーミッションを持ちますが、テーブルの所有者は Company であり、Sally と Joe は Company グループのメンバではないので、彼らがテーブルを参照するときには、修飾を使用する必要があります。

```
SELECT *
FROM company.Customers;
```

この状況を変更するには、次のように Sales グループを Company グループのメンバにします。

```
GRANT GROUP TO Company;
GRANT MEMBERSHIP IN GROUP Company TO Sales;
```

これで、Joe と Sally は、Sales グループのメンバであると同時に間接的に Company グループのメンバになり、修飾子なしでデータを参照できます。したがって、次のコマンドを使えるようになります。

```
SELECT *
FROM Customers;
```

注意

Joe と Sally は company グループのメンバシップ以外のパーミッションを持ちません。company グループには、明示的に付与されたテーブル・パーミッションはありません(company ユーザ ID はテーブルの作成者であり、DBA 権限を持っているので、Salaries のようなテーブルを参照するパーミッションを暗黙のうちに持っています)。したがって、Joe と Sally は、次のコマンドを実行するとエラーになります。

```
SELECT *
FROM Salaries;
SELECT *
FROM company.Salaries;
```

どちらの場合も、Joe と Sally は Salaries テーブルを参照するパーミッションを持っていません。

高度なセキュリティを実現するためのビューとプロシージャの使い方

高レベルのセキュリティが必要なデータベースでは、テーブルに対して直接パーミッションを定義することには限界があります。ユーザに与えたテーブルのパーミッションは、テーブル全体に対して適用されます。ところが、テーブルごとでなくユーザのパーミッションをより厳密に定義する必要がある場合が、数多くあります。次に例を示します。

- employee テーブルにアクセスする必要があるユーザに対して、テーブル中にある個人的な情報にまでアクセスを許可することは望ましくない。
- 営業担当者にセールス・コールの詳細を含むテーブルの更新を許可したいが、担当者自身の部分に対するアクセスだけに制限したい。

これらのケースでは、会社のニーズに応じてパーミッションを調整するためにビューとストアド・プロシージャを使うことができます。この項では、パーミッション管理のためのビューとストアド・プロシージャの使い方について説明します。

参照

- 「ビューの編集」 『SQL Anywhere サーバ - SQL の使用法』
- 「ビューに対するパーミッションの付与」 379 ページ

セキュリティを調整するためにビューを使用する

「**ビュー**」は、ベース・テーブルから選択したローとカラムを含む、計算されたテーブルです。ビューは、ユーザにテーブルの一部分だけに対するアクセス権を与える場合に便利です。その部分はローまたはカラムで定義します。たとえば、ユーザに employee テーブルの Salary カラムが見えないようにしたり、ユーザが自分で作成したローだけを見られるようにしたりできます。

例

Sales Manager は、自分の部署の営業部員に関するデータベースの情報にアクセスする必要があります他部署の従業員の情報にアクセスしなければならない理由はありません。

例として、Sales Manager のユーザ ID を作成してから必要な情報を得るためのビューを作成し、Sales Manager のユーザ ID に適切なパーミッションを与える手順を次に示します。

1. GRANT 文を使用して、新しいユーザ ID を作成します。DBA としてログインし、次のように入力します。

   ```
   CONNECT DBA
   IDENTIFIED by sql;

   GRANT CONNECT
   TO SalesManager
   IDENTIFIED BY sales;
   ```

2. 営業部の従業員だけを見るビューを定義します。


```
CREATE VIEW EmployeeSales AS
  SELECT EmployeeID, GivenName, Surname
  FROM Employees
  WHERE DepartmentID = 200;
```

テーブル参照を所有者で修飾することによって、同じ名前のテーブルの参照と混同されるおそれがなくなります。

3. SalesManager にビューを見るパーミッションを与えます。

```
GRANT SELECT
ON EmployeeSales
TO SalesManager;
```

まったく同じコマンドを使用して、ビューおよびテーブルに対するパーミッションを与えます。

例 2

次の例では、Sales Manager が注文のまとめを確認できるようにするビューを作成します。このビューは、複数のテーブルからの情報を必要とします。

1. ビューを作成します。

```
CREATE VIEW OrderSummary AS
  SELECT OrderDate, Region, SalesRepresentative, CompanyName
  FROM SalesOrders
    KEY JOIN Customers;
```

2. Sales Manager にこのビューを見るパーミッションを与えます。

```
GRANT SELECT
ON OrderSummary
TO SalesManager;
```

3. プロセスが正常に動作したことをチェックするには、ユーザ ID SalesManager に接続し、作成したビューを見ます。

```
CONNECT SalesManager
IDENTIFIED BY sales;
SELECT *
FROM DBA.EmployeeSales;
SELECT *
FROM DBA.OrderSummary;
```

Sales Manager には基本となるテーブルを見るパーミッションは与えられていません。次のコマンドはパーミッション・エラーを起こします。

```
SELECT * FROM GROUPO.Employees;
SELECT * FROM GROUPO.SalesOrders;
```

ビューに対するその他のパーミッション

前述の例では、SELECT パーミッションを調整するためのビューの使用法を説明しました。同じ方法で、INSERT、DELETE、UPDATE の各パーミッションをビューに付与できます。

ビューでデータを修正する方法については、「ビューの使い方」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

セキュリティを調整するためのプロシージャを使用する

ビューはデータへのアクセスを制限しますが、プロシージャはユーザの行動を制限します。ユーザは、プロシージャの対象になるテーブルに対するパーミッションがなくても、プロシージャの EXECUTE パーミッションを持つことができます。「プロシージャに対するパーミッションの付与」 381 ページを参照してください。

厳密なセキュリティ

セキュリティを完全にするには、基本となるテーブルへのアクセスをすべて禁止し、ユーザまたはユーザのグループには、特定のストアド・プロシージャを実行するパーミッションだけを与えます。この方法であれば、データベースのデータの修正方法を厳密に定義できます。

ネストされたオブジェクトの所有権の変更

ビューとプロシージャは、さまざまなユーザが所有する基本のオブジェクトにアクセスできます。たとえば、usera、userb、userc、userd が別々の 4 名のユーザである場合は、userc.viewc から userd.viewd を作成できます。この userc.viewc は、usera.table から作成された userb.viewb をベースにして作成できます。同じように、プロシージャでも、userd.procd は userc.procc を呼び出すことができ、userc.procc は、usera.tablea に挿入できる userb.procb を呼び出すことができます。

ネストされたビューおよびテーブルには、次の DAC (任意アクセス制御) 規則が適用されます。

- ビューを作成するには、ユーザは、そのビューに含まれるすべてのベース・オブジェクト (たとえばテーブルとビュー) に対する SELECT パーミッションが必要です。
- ビューにアクセスするには、ビューの所有者は、基本のテーブルまたはビューに対する適切なパーミッションを GRANT オプションで付与されている必要があります。また、ユーザは、ビューに対する適切なパーミッションを付与されている必要があります。
- WHERE 句を使用して更新するには、SELECT パーミッションと UPDATE パーミッションの両方が必要です。
- ユーザがビュー定義内のテーブルを所有している場合は、そのユーザがビューの所有者ではなく、ビューに対するアクセス権を付与されていなくても、ビューを介してテーブルにアクセスできます。

ネストされたプロシージャには、次の DAC 規則が適用されます。

- プロシージャを作成する場合、ユーザは、基本となるオブジェクト (たとえば、テーブル、ビュー、またはプロシージャ) に対するパーミッションを必要としません。
- プロシージャを実行する場合、プロシージャの所有者は、そのプロシージャが参照するオブジェクトに対する適切なパーミッションを必要とします。
- プロシージャによって参照されるすべてのテーブルをユーザが所有する場合でも、プロシージャに対する EXECUTE パーミッションを付与されていないかぎり、プロシージャを実行してテーブルにアクセスすることはできません。

この動作については、次の例で説明します。

例 1: user1 が table1 を作成し、user2 が table1 について view2 を作成する

- user1 は所有者であるため、常に table1 にアクセスできます。
- user1 は、基本となるテーブルの所有者であるため、常に view2 を介して table1 にアクセスできます。これは、user2 が view2 のパーミッションを user1 に付与しない場合でも該当します。
- user2 が table1 に直接または view2 を介してアクセスできるのは、user1 が table1 のパーミッションを user2 に付与した場合です。
- user3 が table1 にアクセスできるのは、user1 が table1 のパーミッションを user3 に付与した場合です。

- user3 は、user1 が table1 のパーミッションを grant オプションで user2 に付与し、かつ、user2 が view2 のパーミッションを user3 に付与した場合に、view2 を介して table1 にアクセスできます。

例 2: table1 にアクセスする procedure2 を user2 が作成する

- user1 が procedure2 を介して table1 にアクセスできるのは、user2 が procedure2 の EXECUTE パーミッションを user1 に付与した場合です。view2 では、user1 はパーミッションを必要としませんが、上記の場合とは異なるので注意してください。

例 3: user1 が table1 を作成し、user2 が table2 を作成し、user3 が table1 と table2 をジョインする view3 を作成する

- user3 が view3 を介して table1 と table2 にアクセスできるのは、user1 が table1 のパーミッションを user3 に付与し、かつ、user2 が table2 のパーミッションを user3 に付与した場合です。
- user3 が table1 のパーミッションを持っているが table2 のパーミッションを持っていない場合、user3 は view3 を使用できません。table1 のカラムからなるサブセットにもアクセスできません。
- user1 または user2 が view3 を使用できるのは、(a) user1 が table1 のパーミッションを grant オプションとともに user3 に付与し、(b) user2 が table2 のパーミッションを grant オプションとともに user3 に付与し、さらに (c) user3 が view3 のパーミッションを user1 または user2 に付与した場合です。

ユーザ・パーミッションの評価方法

グループによって個々のユーザのパーミッションは複雑になります。たとえば、ユーザ M_Haneef が、あるテーブルに対して SELECT と UPDATE のパーミッションを個人で所有しており、2 つのグループのメンバでもあるとします。一方のグループではテーブルに対するパーミッションをまったく持たず、もう一方では SELECT パーミッションだけを持つとします。この場合、このユーザのパーミッションは一体どうなるでしょうか。

SQL Anywhere は、ユーザ ID が特定のアクションを実行するパーミッションを持っているかどうかを、次の順で判断します。

1. ユーザ ID が DBA 権限を持っている場合、そのユーザはデータベース内ですべての作業を実行できます。
2. DBA 権限がない場合、パーミッションは個別のユーザに与えられたパーミッションによって異なります。ユーザ ID にその作業を実行するパーミッションが付与されていれば、その作業は実行できます。
3. ユーザに対して個別の設定が行われていない場合、パーミッションはそのユーザが属する各グループのパーミッションによって異なります。いずれかのグループがその作業を実行するパーミッションを持っていれば、そのユーザもメンバとしてパーミッションを持っていることになり、その作業を実行できます。

このようにして、パーミッションが設定されている順序に関する問題を最小限に抑えています。

リソース接続使用の管理

ユーザとグループのセットを作成すると、データベースのパーミッションを管理できます。またデータベースのセキュリティと管理によって、個々のユーザが使用できるリソースを制限することもできます。

たとえば、他のユーザのデータベースへの接続速度を落とさないように、1 つの接続がメモリや CPU を大量に使用することを防止できます。

SQL Anywhere には、DBA がリソースの制御に使用できるデータベース・オプションが備わっています。このオプションを「**リソース・ガバナー**」といいます。

オプションの設定

SET OPTION 文を次のように使用して、データベース・オプションを設定します。

SET [**TEMPORARY**] **OPTION** … [*userid*. | **PUBLIC**.]*option-name* = [*option-value*]

管理できるリソース

次のオプションを使用して、リソースを管理できます。

- **max_cursor_count**　接続用のカーソルの数を制限します。「max_cursor_count オプション [データベース]」 464 ページを参照してください。
- **max_statement_count**　接続用の準備文の数を制限します。「max_statement_count オプション [データベース]」 467 ページを参照してください。
- **background_priority**　現在の接続に関する要求が他の接続のパフォーマンスに与える影響を制限します。「background_priority オプション [データベース]」 430 ページを参照してください。

データベース・オプション設定は、グループ構造に継承されません。

参照

- 「データベース・オプション」 405 ページ
- 「SET OPTION 文」『SQL Anywhere サーバ - SQL リファレンス』

カタログのユーザとパーミッション

データベースのシステム・ビューには、データベースの現在のユーザとそのパーミッションに関する情報が含まれています。

システム・ビューの所有者は、特殊なユーザ ID SYS です。ユーザ ID SYS を使用して接続することはできません。

DBA には、すべてのシステム・ビューに対する SELECT アクセス権はありますが、基本となるシステム・テーブルに対する SELECT アクセス権はありません。他のユーザについても、一部のテーブルとビューへのアクセスは制限されます。たとえば、SYS.SYSUSERPERM ビューには、DBA だけがアクセスできます。このテーブルには、データベースのユーザのパーミッションについてのすべての情報と各ユーザ ID の暗号化されたパスワードが格納されています。ただし、SYS.SYSUSERPERMS ビューは、SYS.SYSUSERPERM テーブル内のパスワード以外のすべての情報を含むビューであり、デフォルトでは、すべてのユーザがこのビューに対して SELECT アクセスを持っています。新しいデータベース内で SYS、PUBLIC、DBA、および dbo に対して設定されたすべてのパーミッションとグループ・メンバシップは、自由に変更できます。

次の表に、ユーザ ID、グループ、パーミッションに関する情報を含むシステム・ビューの概要を示します。ユーザ ID SYS はリストされたすべてのビューを所有し、その修飾された名前は SYS.SYSUSERPERM などになります。

これらのビューに対して適切な SELECT クエリを使用すると、すべてのユーザ ID とパーミッションの情報を得られます。

ビュー	デフォルト	内容
SYSCOLAUTH	PUBLIC	SYSCOLPERM の情報を読みやすくしたもの。「SYSCOLAUTH 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSCOLPERM	PUBLIC	GRANT コマンドで与えられる SELECT または UPDATE パーミッションを持つすべてのカラム。「SYSCOLPERM システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
DUMMY	PUBLIC	現在のユーザ ID を知るのに使用できるダミー・テーブル。「DUMMY システム・テーブル」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSGROUP	PUBLIC	各グループのメンバごとに 1 ロー。「SYSGROUP システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSGROUPS	PUBLIC	SYSGROUP の情報を読みやすくしたもの。「SYSGROUPS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

ビュー	デフォルト	内容
SYSPROCAUTH	PUBLIC	SYSPROCPERM の情報を読みやすくしたもの。「SYSPROCAUTH 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSPROCPERM	PUBLIC	各ローには、1 つのプロシージャを使うパーミッションを与えられた 1 人のユーザが含まれる。「SYSPROCPERM システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSTABAUTH	PUBLIC	SYSTABLEPERM の情報を読みやすくしたもの。「SYSTABAUTH 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSTABLEPERM	PUBLIC	GRANT コマンドで付与されるテーブルに関するすべてのパーミッション。「SYSTABLEPERM システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSUSERAUTH	DBA のみ	ユーザ ID 以外のすべての SYSUSERPERM の情報。「SYSUSERAUTH 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSUSERAUTHORITY	PUBLIC	各ユーザ ID に対して付与されている権限。「SYSUSERAUTHORITY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSUSERLIST	PUBLIC	パスワード以外のすべての SYSUSERAUTH の情報。「SYSUSERLIST 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSUSERPERM	DBA のみ	データベース・レベルのパーミッションと各ユーザ ID のパスワード。「SYSUSERPERM 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照。
SYSUSERPERMS	PUBLIC	パスワード以外のすべての SYSUSERPERM の情報。「SYSUSERPERMS 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

第 12 章

データベース・オプション

目次

データベース・オプションの概要

データベース・オプションは、データベースの動作をさまざまな面から制御します。たとえば、次の目的でデータベース・オプションを使用できます。

- **互換性**　SQL Anywhere データベースの動作をどの程度まで Adaptive Server Enterprise に近づけるかを指定し、SQL が SQL/2003 に準拠しない場合にエラーを発生させるかどうかを設定できます。
- **エラー処理**　ゼロ除算やオーバフローなどのエラーが起こったときの処理方法を制御できます。
- **同時実行性とトランザクション**　同時実行性の程度と、COMMIT 動作の詳細を制御できます。

オプションの設定

オプションは SET OPTION 文を使用して設定します。一般的な構文は、次のとおりです。

SET [**EXISTING**] [**TEMPORARY**] **OPTION**
[*userid*. | **PUBLIC**.]*option-name* = [*option-value*]

特定のユーザまたはグループにのみオプションを設定するには、ユーザ ID かグループ名を指定します。すべてのユーザは PUBLIC グループに属します。ユーザ ID またはグループを指定しない場合、SET OPTION 文を発行した、現在ログオンしているユーザ ID にオプション変更が適用されます。

どのようなオプションでも、ユーザ定義であるかどうかにかかわらず、パブリック設定がなければユーザ固有の値を割り当てることはできません。データベース・サーバでは、ユーザ定義オプションへの TEMPORARY 値の設定はサポートされていません。

たとえば、次の文では、オプション変更を発行したユーザが DBA の場合、ユーザ DBA にその変更が適用されます。

```
SET OPTION login_mode = Integrated;
```

次の文は、すべてのユーザが属するユーザ・グループである PUBLIC ユーザ ID に変更を適用します。

```
SET OPTION Public.login_mode = Standard;
```

option-value を省略すると、指定したオプション設定がデータベースから削除されます。これが個人的なオプション設定の場合は、値は PUBLIC 設定に戻ります。TEMPORARY オプションを削除すると、オプション設定は永久設定に戻ります。

「SET OPTION 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

警告
カーソルからローをフェッチしている間のオプション設定の変更はサポートされていません。このような変更を行うと、結果の信頼性が損なわれる可能性があるからです。たとえば、カーソルからフェッチしている間に date_format 設定を変えると、結果セットのロー同士の日付フォーマットが異なってしまいます。ローをフェッチしている間にオプション設定を変更しないでください。

注意
トルコ語照合を使用しているデータベースや大文字と小文字が区別されるデータベースでは、オプション名の大文字と小文字が正しくないと、SYSOPTION のクエリや次のようなクエリでは、どのローとも一致しない可能性があります。
SELECT * FROM sa_conn_properties() WHERE propname = 'BLOCKING';
オプション名における大文字と小文字の正しい対応の詳細については、「アルファベット順のオプション・リスト」 421 ページを参照してください。

データベース・オプションのスコープと継続期間

オプションには、パブリック、ユーザ、テンポラリの 3 レベルのスコープを設定できます。

テンポラリ・オプションは、ユーザ設定とパブリック設定より優先されます。ユーザ・レベルのオプションは、パブリック設定より優先されます。現在のユーザに対してユーザ・レベルのオプションを設定すると、対応するテンポラリ・オプションも設定されます。

オプションの中には (COMMIT 動作など)、スコープがデータベース全体に及ぶものがあります。これらのオプションの設定には、DBA パーミッションが必要です。その他のオプション (isolation_level など) は、現在の接続のみに適用され、特別なパーミッションは必要ありません。

オプション設定への変更がいつ有効になるかは、オプションによって異なります。recovery_time などのグローバル・オプションの変更は、次にデータベースを起動したときに有効になります。

現在の接続にのみ影響を与えるオプションは、通常すぐに有効になります。オプションの設定は、たとえばトランザクションの途中でも変更できます。ただし、*カーソルが開いているときにオプションを変更すると、結果の信頼性が低くなる*可能性があります。たとえば、カーソルが開いているときに date_format を変更しても、次のローのフォーマットは変わりません。カーソルが取り出される方法によっては、変更がユーザに伝わるのは、いくつか後のローになる場合があります。

パブリック・オプションの設定

PUBLIC ユーザ ID に対してオプションを設定するには、DBA 権限が必要です。

PUBLIC ユーザ ID に対するオプションの値を変更すると、独自の数値を設定していない全ユーザについて、そのオプションの永久値が変更されます。ただし、そのオプションがすでに PUBLIC ユーザ ID に設定されていないかぎり、個々のユーザ ID にオプション値は設定されません。

PUBLIC ユーザにかぎり設定されたオプションの中には、変更された設定を CONNECTION_PROPERTY 関数を通じてユーザが参照できなくても、既存の接続に対して即座に有効になるものがあります。このような例には、global_database_id オプションがあります。このような理由から、他のユーザがデータベースに接続している間は PUBLIC-only オプションを変更しないでください。

テンポラリ・オプションの設定

SET OPTION 文に TEMPORARY キーワードを追加すると、変更の適用期間を変更できます。通常、オプションの変更は永久的です。SET OPTION 文を使って明示的に変更されるまで変わりません。

SET TEMPORARY OPTION 文を実行した場合、新しいオプション値は現在の接続にかぎり、接続の期間のみ有効になります。

SET TEMPORARY OPTION を使用して PUBLIC オプションを設定すると、データベースの実行中はその変更が継続します。データベースが停止した際に、PUBLIC ユーザ ID のテンポラリ・オプションはその永久値に戻ります。

PUBLIC ユーザ ID のテンポラリ・オプションを設定すると、セキュリティ上の利点があります。たとえば、login_mode オプションが有効なときは、データベースは実行中のシステムのログイン・セキュリティに依存します。このオプション設定を一時的に有効にすると、Windows ドメインのセキュリティに依存しているデータベースは、データベースが停止し、ローカル・コンピュータにコピーされるといった場合でも、危険にさらされることはありません。この場合、login_mode オプションは、永久値の Standard に戻ります。このモードでは、統合化ログインを使用できません。

SQL 文の設定オプション

INSERT、UPDATE、DELETE、SELECT 文には OPTION 句があるため、実体化ビュー (Materialized View) にこの文が影響する方法とクエリを最適化する方法を指定できます。この句を使用すると、該当する文に対してのみ、パブリック・オプションやテンポラリ・オプションの設定よりも優先されるオプション設定を指定できます。OPTION 句では、次のオプションの設定を変更できます。

- isolation_level
- max_query_tasks
- optimization_goal
- optimization_level
- optimization_workload

オプション設定の検索

さまざまな方法でオプション設定のリストや個別のオプションの値を入手できます。

オプション値のリストを取得する

- 現在の接続のオプション設定は、「**接続プロパティ**」のサブセットとして取得できます。sa_conn_properties システム・プロシージャを使用すると、すべての接続プロパティをリストできます。


```
CALL sa_conn_properties;
```

このリストをアルファベット順に並べ替えるには、次の文を実行します。

```
SELECT *
FROM sa_conn_properties()
ORDER BY PropName;
```

結果をフィルタしたり、または名前以外の順番で並べたりしたい場合、SELECT 文も使用できます。次に例を示します。

```
SELECT *
FROM sa_conn_properties()
WHERE PropDescription LIKE '%cache%'
ORDER BY PropNum;
```

「sa_conn_properties システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- Interactive SQL では、引数なしで SET 文を使用すると、現在のオプション設定がリストされます。

  ```
  SET;
  ```

- Sybase Central で、データベースを選択し、[ファイル] ► [オプション] を選択します。
- SYSOPTIONS システム・ビューで、次のクエリを使用します。

  ```
  SELECT *
  FROM SYSOPTIONS;
  ```

  このクエリでは、すべての PUBLIC 値と、明示的に設定された USER 値が表示されます。

個別のオプション値を取得する

CONNECTION_PROPERTY システム関数を使用して、個々の設定を取得できます。たとえば次の文では、ansi_integer_overflow オプションの値が表示されます。

```
SELECT CONNECTION_PROPERTY ('ansi_integer_overflow');
```

「CONNECTION_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

オプションの初期設定

SQL Anywhere への接続には、TDS プロトコル (Open Client 接続と jConnect JDBC 接続) と SQL Anywhere プロトコル (ODBC と Embedded SQL) のどちらも使用できます。

TDS プロトコルと SQL Anywhere 固有のプロトコルの両方を使用する場合、ストアド・プロシージャを使用して初期設定を行うことができます。SQL Anywhere の出荷時には、この方法を使用して、デフォルトの Adaptive Server Enterprise の動作に合わせ、Open Client 接続と jConnect 接続を設定しています。

最初の設定は、login_procedure オプションによって制御されます。このオプションは、ユーザが接続したときに実行されるストアド・プロシージャを指定します。デフォルトの設定では、

sp_login_environment システム・プロシージャが使用されます。この動作は、必要に応じて変更できます。

sp_login_environment は、接続が TDS を介して行われているかどうかをチェックします。そうである場合は、sp_tsql_environment プロシージャを呼び出して、いくつかのオプションに現在の接続に合った新しいデフォルト値を設定します。

参照

- 「login_procedure オプション [データベース]」 460 ページ
- 「sp_login_environment システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
- 「sp_tsql_environment システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』

オプション設定の削除

option-value を省略すると、指定したオプション設定がデータベースから削除されます。これが個人的なオプション設定の場合は、値は PUBLIC 設定に戻ります。TEMPORARY オプションを削除すると、オプション設定は永久設定に戻ります。

たとえば、次の文は、ansi_integer_overflow オプションをデフォルト値にリセットします。

```
SET OPTION ansi_integer_overflow =;
```

「SET OPTION 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

オプション分類

SQL Anywhere には、さまざまなオプションが用意されています。いくつかの一般的なクラスに分類すると便利です。オプションのクラスは次のとおりです。

- 「データベース・オプション」 410 ページ
- 「互換性オプション」 416 ページ
- 「SQL Remote オプション」 420 ページ
- 「Interactive SQL オプション」 607 ページ

データベース・オプション

この項には、すべてのデータベース・オプションの一覧を掲載します。

オプション	値	デフォルト
「allow_snapshot_isolation オプション [データベース]」 422 ページ	On、Off	Off
「auditing オプション [データベース]」 428 ページ	On、Off	Off

オプション	値	デフォルト
「auditing_options オプション [データベース]」 429 ページ	予約	予約
「background_priority オプション [データベース]」 430 ページ	On、Off	Off
「blocking オプション [データベース]」 431 ページ	On、Off	On
「blocking_timeout オプション [データベース]」 431 ページ	整数 (ミリ秒)	0
「checkpoint_time オプション [データベース]」 432 ページ	分	60
「cis_option オプション [データベース]」 433 ページ	0, 7	0
「cis_rowset_size オプション [データベース]」 433 ページ	整数	50
「collect_statistics_on_dml_updates オプション [データベース]」 434 ページ	On、Off	On
「conn_auditing オプション [データベース]」 435 ページ	On、Off	On
「connection_authentication オプション [データベース]」 436 ページ	文字列	空の文字列
「cooperative_commit_timeout オプション [データベース]」 438 ページ	整数 (ミリ秒)	250
「cooperative_commits オプション [データベース]」 439 ページ	On、Off	On
「database_authentication [データベース]」 440 ページ	文字列	空の文字列
「debug_messages オプション [データベース]」 443 ページ	On、Off	Off
「dedicated_task オプション [データベース]」 444 ページ	On、Off	Off
「default_dbspace オプション [データベース]」 444 ページ	文字列	空の文字列 (システム DB 領域を使用)
「default_timestamp_increment オプション [データベース] [Mobile Link クライアント]」 445 ページ	整数 (1 ー 1000000 マイクロ秒)	1

オプション	値	デフォルト
「delayed_commit_timeout オプション [データベース]」 446 ページ	整数 (ミリ秒)	500
「delayed_commits オプション [データベース]」 446 ページ	On、Off	Off
「encrypt_aes_random_iv オプション [データベース]」 448 ページ	On、Off	On
「exclude_operators オプション [データベース]」 449 ページ	予約	予約
「extended_join_syntax オプション [データベース]」 449 ページ	On、Off	On
「first_day_of_week オプション [データベース]」 450 ページ	1, 2, 3, 4, 5, 6, 7	7 (1 週間は日曜日から始まる)
「for_xml_null_treatment オプション [データベース]」 452 ページ	Empty、Omit	Omit
「force_view_creation オプション [データベース]」 452 ページ	予約	予約
「global_database_id オプション [データベース]」 453 ページ	整数	2147483647
「http_session_timeout オプション [データベース]」 453 ページ	整数	30
「integrated_server_name オプション [データベース]」 454 ページ	文字列	NULL
「java_location オプション [データベース]」 456 ページ	文字列	空の文字列
「java_main_userid オプション [データベース]」 457 ページ	文字列	デフォルト DBA ユーザ
「java_vm_options オプション [データベース]」 457 ページ	文字列	空の文字列
「log_deadlocks オプション [データベース]」 458 ページ	On、Off	Off
「login_mode オプション [データベース]」 459 ページ	Standard、Integrated、Kerberos、Mixed (旧式)	Standard
「login_procedure オプション [データベース]」 460 ページ	文字列	sp_login_environment

オプション	値	デフォルト
「materialized_view_optimization オプション [データベース]」 462 ページ	Disabled、Fresh、Stale、*N* { Minute [s] | Hour[s] | Day[s] | Week[s] | Month [s] }	Stale
「max_client_statements_cached オプション [データベース]」 463 ページ	整数	10
「max_cursor_count オプション [データベース]」 464 ページ	整数	50
「max_plans_cached オプション [データベース]」 465 ページ	整数	20
「max_query_tasks オプション [データベース]」 466 ページ	整数	0
「max_recursive_iterations オプション [データベース]」 467 ページ	整数	100
「max_statement_count オプション [データベース]」 467 ページ	0 以上の整数	50
「max_temp_space オプション [データベース]」 469 ページ	整数 [k|m|g|p]	0
「min_password_length オプション [データベース]」 470 ページ	0 以上の整数	0 文字
「odbc_describe_binary_as_varbinary [データベース]」 471 ページ	On、Off	Off
「odbc_distinguish_char_and_varchar オプション [データベース]」 472 ページ	On、Off	Off
「oem_string オプション [データベース]」 473 ページ	文字列 (最大 128 バイト)	空の文字列
「on_charset_conversion_failure オプション [データベース]」 475 ページ	Ignore、Warning、Error	Ignore
「optimization_goal オプション [データベース]」 477 ページ	First-row または All-rows	All-rows
「optimization_level オプション [データベース]」 478 ページ	0−15	9
「optimization_workload オプション [データベース]」 479 ページ	Mixed、OLAP	Mixed

オプション	値	デフォルト
「pinned_cursor_percent_of_cache オプション [データベース]」 481 ページ	整数 (0 ～ 100)	10
「post_login_procedure [データベース]」 481 ページ	文字列	空の文字列
「precision オプション [データベース]」 483 ページ	整数 (1 ～ 127)	30
「prefetch オプション [データベース]」 483 ページ	Off、Conditional、Always	Conditional
「preserve_source_format オプション [データベース]」 484 ページ	On、Off	On
「prevent_article_pkey_update オプション [データベース] [Mobile Link クライアント]」 485 ページ	On、Off	On
「read_past_deleted オプション [データベース]」 487 ページ	On、Off	On
「recovery_time オプション [データベース]」 488 ページ	整数 (分)	2
「remote_idle_timeout オプション [データベース]」 488 ページ	整数 (秒)	15
「request_timeout オプション [データベース]」 490 ページ	整数 (0 ～ 86400 秒)	0
「return_date_time_as_string オプション [データベース]」 491 ページ	On、Off	Off
「rollback_on_deadlock [データベース]」 492 ページ	On、Off	On
「row_counts オプション [データベース]」 493 ページ	On、Off	Off
「scale オプション [データベース]」 494 ページ	整数 (0 ～ 127 の範囲で、かつ precision データベース・オプションの値より小さいことが必要)	6
「secure_feature_key [データベース]」 494 ページ	文字列	NULL

オプション	値	デフォルト
「sort_collation オプション [データベース]」 495 ページ	Internal、collation_name、collation_id	Internal
「subsume_row_locks オプション [データベース]」 501 ページ	On、Off	On
「suppress_tds_debugging オプション [データベース]」 502 ページ	On、Off	Off
「synchronize_mirror_on_commit オプション [データベース]」 502 ページ	On、Off	Off
「tds_empty_string_is_null オプション [データベース]」 503 ページ	On、Off	Off
「temp_space_limit_check オプション [データベース]」 503 ページ	On、Off	On
「time_zone_adjustment オプション [データベース]」 505 ページ	整数、引用符で囲まれた負の整数、先頭に + または - が付いた時刻 (時間と分) を表す文字列 (引用符で囲まれたもの) のいずれか	クライアントの接続タイプに応じて、クライアントまたはサーバのタイム・ゾーンのいずれかによって設定される
「truncate_timestamp_values オプション [データベース] [Mobile Link クライアント]」 507 ページ	On、Off	Off
「truncate_with_auto_commit オプション [データベース]」 508 ページ	On、Off	On
「updatable_statement_isolation オプション [データベース]」 510 ページ	0, 1, 2, 3	0
「update_statistics オプション [データベース]」 510 ページ	On、Off	On
「user_estimates オプション [データベース]」 511 ページ	Enabled、Disabled、Override-Magic	Override-Magic
「uuid_has_hyphens オプション [データベース]」 512 ページ	On、Off	On
「verify_password_function オプション [データベース]」 513 ページ	文字列	空の文字列
「wait_for_commit オプション [データベース]」 515 ページ	On、Off	Off

オプション	値	デフォルト
「webservice_namespace_host オプション [データベース]」 515 ページ	NULL、hostname-string	NULL

互換性オプション

次のオプションを使用すると、SQL Anywhere の動作に Adaptive Server Enterprise との互換性を持たせたり、両方の旧バージョンの動作をサポートしたり、ISO SQL/2003 動作を使用可能にしたりできます。

Adaptive Server Enterprise との互換性をさらに高めるために、SQL Anywhere の SET OPTION 文の代わりに Transact-SQL の SET 文を使用して、いくつかのオプションを現在の接続の間だけ設定できます。「SET 文 [T-SQL]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

デフォルトの設定値

これらのオプションのデフォルト設定は、Adaptive Server Enterprise のデフォルト設定と一部異なります。SQL Anywhere データベースと Adaptive Server Enterprise データベース間の互換性を確保するためには、この項に挙げた個々の互換性オプションを明示的に設定する必要があります。

Open Client または JDBC インタフェースを使用して接続が行われた場合は、Adaptive Server Enterprise との互換性のために、現在の接続にいくつかのオプションが明示的に設定されています。次の表は、これらのオプションを示しています。

Open Client と JDBC の Adaptive Server Enterprise との接続の互換性を維持するためのオプション

オプション	設定
allow_nulls_by_default	Off
ansi_blanks	Off
ansinull	On
chained	Off
continue_after_raiserror	On
date_format	YYYY-MM-DD
date_order	MDY
escape_character	Off
float_as_double	On
isolation_level	1

オプション	設定
on_tsql_error	jConnect 接続の場合は Continue
quoted_identifier	Off
time_format	HH:NN:SS.SSS
timestamp_format	YYYY-MM-DD HH:NN:SS.SSS
tsql_hex_constant	On
tsql_outer_joins	Off
tsql_variables	On

Transact-SQL と SQL/2003 の互換性オプション

次の表は、互換性オプション、指定可能な値、デフォルト設定のリストです。

オプション	値	デフォルト
「allow_nulls_by_default オプション [互換性]」 421 ページ	On、Off	On
「ansi_blanks オプション [互換性]」 423 ページ	On、Off	Off
「ansi_close_cursors_on_rollback オプション [互換性]」 424 ページ	On、Off	Off
「ansi_integer_overflow オプション [互換性]」 424 ページ	On、Off	On
「ansi_permissions オプション [互換性]」 424 ページ	On、Off	On
「ansi_substring オプション [互換性]」 425 ページ	On、Off	On
「ansi_update_constraints オプション [互換性]」 426 ページ	Off、Cursors、Strict	Cursors
「ansinull オプション [互換性]」 427 ページ	On、Off	On
「automatic_timestamp オプション [互換性]」 429 ページ	On、Off	Off
「chained オプション [互換性]」 432 ページ	On、Off	On
「close_on_endtrans オプション [互換性]」 434 ページ	On、Off	On

オプション	値	デフォルト
「continue_after_raiserror オプション [互換性]」 437 ページ	On、Off	On
「conversion_error オプション [互換性]」 438 ページ	On、Off	On
「date_format オプション [互換性]」 441 ページ	文字列	YYYY-MM-DD
「date_order オプション [互換性]」 443 ページ	MDY、YMD、DMY	YMD
「divide_by_zero_error オプション [互換性]」 448 ページ	On、Off	On
「escape_character オプション [互換性]」 449 ページ	予約	予約
「fire_triggers オプション [互換性]」 450 ページ	On、Off	On
「float_as_double オプション [互換性]」 451 ページ	On、Off	Off
「isolation_level オプション [互換性]」 455 ページ	0, 1, 2, 3	0
「nearest_century オプション [互換性]」 470 ページ	整数 (0 ー 100)	50
「non_keywords オプション [互換性]」 471 ページ	文字列 (カンマで区切られたキーワード・リスト)	空の文字列 (オフになっているキーワードはない)
「on_tsql_error オプション [互換性]」 475 ページ	Stop、Conditional、Continue	Conditional
「optimistic_wait_for_commit オプション [互換性]」 476 ページ	On、Off	Off
「percent_as_comment オプション [互換性]」 480 ページ	On、Off	On
「query_plan_on_open オプション [互換性]」 486 ページ	On、Off	Off
「quoted_identifier オプション [互換性]」 487 ページ	On、Off	On
「ri_trigger_time オプション [互換性]」 492 ページ	Before、After	After

オプション	値	デフォルト
「sql_flagger_error_level オプション [互換性]」 496 ページ	Off、SQL:1992/Entry、SQL:1992/Intermediate、SQL:1992/Full、SQL:1999/Core、SQL:1999/Package、SQL:2003/Core、SQL:2003/Package、Ultralite	Off
「sql_flagger_warning_level オプション [互換性]」 497 ページ	Off、SQL:1992/Entry、SQL:1992/Intermediate、SQL:1992/Full、SQL:1999/Core、SQL:1999/Package、SQL:2003/Core、SQL:2003/Package、Ultralite	Off
「string_rtruncation オプション [互換性]」 500 ページ	On、Off	On
「time_format オプション [互換性]」 504 ページ	文字列	HH:NN:SS.SSS
「timestamp_format オプション [互換性]」 505 ページ	文字列	YYYY-MM-DD HH:NN:SS.SSS
「tsql_hex_constant オプション [互換性]」 509 ページ	On、Off	On
「tsql_outer_joins オプション [互換性]」 509 ページ	On、Off	Off
「tsql_variables オプション [互換性]」 509 ページ	On、Off	Off

同期オプション

次のデータベース・オプションは、Mobile Link 同期クライアントとして使用する SQL Anywhere データベースの設定に使用できます。

オプション	値	デフォルト
「default_timestamp_increment オプション [データベース] [Mobile Link クライアント]」 445 ページ	整数 (1 ー 1000000 マイクロ秒)	1
「delete_old_logs オプション [Mobile Link クライアント] [SQL Remote] [Replication Agent]」 447 ページ	On、Off、Delay、*n* days	Off
「prevent_article_pkey_update オプション [データベース] [Mobile Link クライアント]」 485 ページ	On、Off	On
「truncate_timestamp_values オプション [データベース] [Mobile Link クライアント]」 507 ページ	On、Off	Off

SQL Remote オプション

SQL Remote のレプリケーション動作を制御するために、次のオプションが用意されています。

オプション	値	デフォルト
「blob_threshold オプション [SQL Remote]」 430 ページ	整数 (キロバイト)	256
「compression オプション [SQL Remote]」 435 ページ	-1 ー 9 の整数	6
「delete_old_logs オプション [Mobile Link クライアント] [SQL Remote] [Replication Agent]」 447 ページ	On、Off、Delay、*n* days	Off
「external_remote_options [SQL Remote]」 450 ページ	On、Off	Off
「qualify_owners オプション [SQL Remote]」 486 ページ	On、Off	On
「quote_all_identifiers オプション [SQL Remote]」 486 ページ	On、Off	Off
「replication_error オプション [SQL Remote]」 489 ページ	ストアド・プロシージャ名	(プロシージャなし)
「replication_error_piece オプション [SQL Remote]」 490 ページ	ストアド・プロシージャ名	(プロシージャなし)
「save_remote_passwords オプション [SQL Remote]」 493 ページ	On、Off	On

オプション	値	デフォルト
「sr_date_format オプション [SQL Remote]」 498 ページ	*日付文字列*	*YYYY/MM/DD*
「sr_time_format オプション [SQL Remote]」 499 ページ	*時刻文字列*	*HH:NN:SS.SSSSSS*
「sr_timestamp_format [SQL Remote]」 500 ページ	*タイムスタンプ文字列*	*YYYY/MM/DD HH:NN:SS.SSSSSS*
「subscribe_by_remote オプション [SQL Remote]」 501 ページ	On、Off	On
「verify_all_columns オプション [SQL Remote]」 512 ページ	On、Off	Off
「verify_threshold オプション [SQL Remote]」 514 ページ	整数 (バイト)	1000

Replication Agent のオプション

Replication Agent のレプリケーション動作を制御するために、次のオプションが用意されています。

オプション	値	デフォルト
「delete_old_logs オプション [Mobile Link クライアント] [SQL Remote] [Replication Agent]」 447 ページ	On、Off、Delay、*n* days	Off
「replicate_all オプション [Replication Agent]」 489 ページ	On、Off	Off

アルファベット順のオプション・リスト

この項では、オプションをアルファベット順に説明します。

allow_nulls_by_default オプション [互換性]

NULL か NOT NULL かを指定しないで作成された新規カラムが、NULL 値を含むのを許可するかどうかを制御します。

指定可能な値

On、Off

デフォルト

On

Open Client 接続と jConnect 接続の場合は Off

備考

allow_nulls_by_default オプションは、Transact-SQL との互換性を保つために実装されています。「Transact-SQL との互換性を維持するためのオプション設定」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

allow_snapshot_isolation オプション [データベース]

スナップショット・アイソレーションを有効または無効にします。

指定可能な値

On、Off

デフォルト

Off

スコープ

PUBLIC グループのみに設定できます。このオプションを設定するには、DBA 権限が必要です。

備考

このオプションは、データベースのスナップショット・アイソレーションを有効にするかどうかを指定します。このオプションを On に設定すると、トランザクションがスナップショット・アイソレーションを使用した場合、データベース・サーバは更新されたローの元のバージョンをテンポラリ・ファイルに記録します。

トランザクションが実行中である場合は、allow_snapshot_isolation オプションの設定を変更しても、新しい設定が直ちに有効になるわけではありません。このオプションの設定を Off から On に変更しても、その時点で実行中のトランザクションが*完了した後*でなければスナップショットは使用できません。このオプションの設定を On から Off に変更した場合は、未完了のスナップショットが完了してからバージョン情報の収集が停止され、新たなスナップショットは開始されません。

特定のデータベースについて現在のスナップショット・アイソレーションの設定を確認するには、SnapshotIsolationState データベース・プロパティの値を問い合わせます。

```
SELECT DB_PROPERTY ( 'SnapshotIsolationState' );
```

SnapshotIsolationState プロパティの値は、次のいずれかです。

- **On**　データベースでスナップショット・アイソレーションが有効になっている。
- **Off**　データベースでスナップショット・アイソレーションが無効になっている。
- **in_transition_to_on**　現在のトランザクションの完了後にスナップショット・アイソレーションが有効となる。

- **in_transition_to_off**　現在のトランザクションの完了後にスナップショット・アイソレーションが無効となる。

参照

- 「isolation_level オプション [互換性]」 455 ページ
- 「updatable_statement_isolation オプション [データベース]」 510 ページ
- 「スナップショット・アイソレーション」『SQL Anywhere サーバ - SQL の使用法』
- 「独立性レベルと一貫性」『SQL Anywhere サーバ - SQL の使用法』
- 「スナップショット・アイソレーションの有効化」『SQL Anywhere サーバ - SQL の使用法』

例

次の文は、データベースのスナップショット・アイソレーションを有効にします。

```
SET OPTION PUBLIC.allow_snapshot_isolation = 'On';
```

ansi_blanks オプション [互換性]

文字データがクライアント・サイドでトランケートされるときの動作を制御します。

指定可能な値

On、Off

デフォルト

Off

備考

ansi_blanks オプションが効力を持つのは、データベースが文字列の比較で後続ブランクを無視し、フェッチした文字列を文字配列に埋め込む場合に限られます。これは、*N* の値が *M* 以上の場合に、データ型 CHAR(*N*) の値が C char(*M*) 変数に読み込まれるたびトランケーション・エラーを強制的に発生させます。ansi_blanks が Off に設定されていると、トランケーション・エラーは、少なくともブランク以外の文字がトランケートされた場合にのみ発生します。

Embedded SQL で ansi_blanks オプションが On に設定されている場合、データ型 DT_STRING の値を挿入するときには、sqllen フィールドを値が保存されているバッファの長さ (最低でも値の長さに末尾の NULL 文字を加えた長さ) に設定する必要があります。ansi_blanks が Off の場合、長さは NULL 文字の位置でのみ決定されます。ansi_blanks オプションの値は、接続が確立されたときに決定されます。接続が確立された後で ansi_blanks オプションの値を変更しても、sqllen Embedded SQL の動作には影響しません。

ブランク埋め込みが有効になっているデータベースでは、このオプションの設定によって、フェッチする式が CHAR または NCHAR であり (VARCHAR または NVARCHAR ではない)、ホスト変数 char または nchar (VARCHAR または NVARCHAR ではない) に格納される場合に、トランケーション警告をクライアントに送信するかどうかが決まります。これらの条件が該当し、ホスト変数が小さすぎるために、フェッチされた式の最大長までブランクが埋め込まれることにより、変数に式を格納できなくなる場合は、トランケーション警告が発生し、フェッチされた式の最大長までブランクが埋め込まれた場合に式の格納に必要となる最小バイト数がインジケータに挿入さ

れます。式が CHAR(N) または NCHAR(N) である場合、返される値の文字セット変換と文字長セマンティクスが考慮され、インジケータが N 以外の値に設定されることがあります。

ansi_close_cursors_on_rollback オプション [互換性]

WITH HOLD 句で開いたカーソルを、ROLLBACK を実行するときに閉じるかどうかを制御します。

指定可能な値

On、Off

デフォルト

Off

備考

ドラフトの SQL/3 標準では、トランザクションをロールバックするときにすべてのカーソルが閉じている必要があります。デフォルトでは、ロールバックした場合、SQL Anywhere は、WITH HOLD 句なしで開いたカーソルだけを閉じます。このオプションを使うと、すべてのカーソルを強制的に閉じることができます。

close_on_endtrans オプションは、ansi_close_cursors_on_rollback オプションを上書きします。

参照

- 「close_on_endtrans オプション [互換性]」 434 ページ

ansi_integer_overflow オプション [互換性]

計算式が整数のオーバフロー・エラーを起こしたときに発生する事象を制御します。

指定可能な値

On、Off

デフォルト

On

備考

ISO SQL/2003 規格では、整数のオーバフローは SQLSTATE = 22003 - **オーバーフロー**のエラーになります。SQL Anywhere の動作は、以前とは異なります。ansi_integer_overflow オプションを Off に設定し、ソフトウェアの旧バージョンと互換性を持たせることができます。

ansi_permissions オプション [互換性]

DELETE と UPDATE 文のパーミッションのチェックを制御します。

指定可能な値

On、Off

デフォルト

On

スコープ

PUBLIC グループのみに設定できます。すぐに有効になります。このオプションを設定するには、DBA 権限が必要です。

備考

ansi_permissions を On にすると、DELETE 文と UPDATE 文に対する SQL/2003 パーミッション要求がチェックされます。Adaptive Server Enterprise では、このオプションのデフォルト値は Off です。次の表はこの違いを説明しています。

SQL 文	ansi_permissions が OFF の時に必要なパーミッション	ansi_permissions が ON の時に必要なパーミッション
UPDATE	値が設定されているカラムでの UPDATE パーミッション	値が設定されているカラムでの UPDATE パーミッション WHERE 句に指定されたすべてのカラムでの SELECT パーミッション SET 句の右側に指定されたすべてのカラムでの SELECT パーミッション
DELETE	テーブルでは DELETE パーミッション	テーブルでは DELETE パーミッション WHERE 句に指定されたすべてのカラムでの SELECT パーミッション

ansi_permissions オプションは、PUBLIC グループのみに設定できます。個人的な設定は許可されません。

ansi_substring オプション [互換性]

start パラメータまたは length パラメータに負の値が設定された場合の SUBSTRING (SUBSTR) 関数の動作を制御します。

指定可能な値

Off、On

デフォルト

On

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

ansi_substring オプションを On に設定した場合、SUBSTRING 関数は ANSI/ISO SQL/2003 と同じ動作をします。開始オフセットが負または 0 の場合は、文字列の左側が文字以外で埋められているかのように扱われ、このときに負の長さが指定されるとエラーになります。

このオプションを Off に設定すると、SUBSTRING 関数は SQL Anywhere の以前のリリースと同じ動作となります。つまり、負の開始オフセットは文字列の末尾からのオフセットを意味し、負の長さは開始オフセットから *length* 文字左側の位置で部分文字列が終わることを意味します。また、開始オフセット 0 を使用した場合、開始オフセット 1 と同じ結果となります。

このオプションの設定は、BYTE_SUBSTR 関数の動作には影響しません。SUBSTRING 関数では、開始オフセットを正でない値に設定したり負の長さを指定したりしないことをおすすめします。可能なかぎり、SUBSTRING 関数の代わりに LEFT 関数または RIGHT 関数を使用してください。

参照

- 「SUBSTRING 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』
- 「LEFT 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』
- 「RIGHT 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』

例

次の例は、ansi_substring オプションの設定によって SUBSTRING 関数の戻り値がどのように変わるかを示しています。

```
SUBSTRING('abcdefgh',-2,4);
  ansi_substring = Off ==> 'gh' // substring starts at second-last character
  ansi_substring = On  ==> 'a'  // takes the first 4 characters of
                           // ???abcdefgh and discards all ?

SUBSTRING('abcdefgh',4,-2);
  ansi_substring = Off ==> 'cd'
  ansi_substring = On  ==> value -2 out of range for destination

SUBSTRING('abcdefgh',0,4);
  ansi_substring = Off ==> 'abcd'
  ansi_substring = On  ==> 'abc'
```

ansi_update_constraints オプション [互換性]

更新可能な範囲を制御します。

指定可能な値

Off、Cursors、Strict

デフォルト

Cursors

備考

SQL Anywhere には、ANSI SQL 規格では許可されていない更新が可能な言語拡張機能がいくつか含まれています。これらの言語拡張機能を利用して、強力かつ効率的に更新を行うことができます。ただし、直感的に予期しない動作が起きる場合もあります。ユーザのアプリケーションがこれらの言語拡張機能の動作を予期して設計されていない場合、この動作によって、更新内容の消失などの異常事態が発生することがあります。

ansi_update_constraints では、更新を SQL/2003 規格で許可される内容に限定するかどうかを制御します。

このオプションを Strict に設定すると、次の更新は許可されません。

- JOINS を含んだカーソルの更新
- ORDER BY 句に表示されるカラムの更新
- FROM 句は、UPDATE 文では使用できません。

このオプションを Cursors に設定した場合は、カーソルにだけ同じ制限が加えられます。カーソルが FOR UPDATE または FOR READ ONLY で開かれていない場合は、データベース・サーバは SQL/2003 規格に基づいて、更新可能かどうかを選択します。ansi_update_constraints オプションが Cursors または Strict の場合、ORDER BY 句を含むカーソルは、デフォルトの FOR READ ONLY になります。それ以外の場合は、引き続きデフォルトの FOR UPDATE になります。

参照

- 「UPDATE 文」『SQL Anywhere サーバ - SQL リファレンス』

ansinull オプション [互換性]

NULL 値の解釈を制御します。

指定可能な値

On、Off

デフォルト

On

備考

このオプションは、主に Transact-SQL (Adaptive Server Enterprise) との互換性を保つために実装されています。ansinull オプションは、NULL 定数を持つ比較述部の結果に影響します。また、NULL 値でグループ化されたクエリに対して発行される警告にも影響します。

ansinull を On にすると、ANSI 3 値的論理が WHERE 句、HAVING 句、または On 状態におけるすべての述部比較で使用されます。**=** または **!=** を使用した NULL との比較はすべて不定と評価されます。

ansinull を Off に設定すると、SQL Anywhere は次の 4 つの条件に対して 2 値的論理を使用します。

expr = NULL

expr > != NULL

expr = @var // @var はプロシージャ変数またはホスト変数

expr != @var

いずれの場合も、述部が true または false (不定でない) と評価します。このような比較では、NULL 値は各ドメインで特別な値として処理され、2 つの NULL 値の等号 (=) 比較は true になります。式 *expr* は、相対的に単純な式で、カラム、変数、リテラルのみを参照し、サブクエリや関数を許可しないようにしてください。

ansinull を On に設定した場合、NULL 値が少なくとも 1 つ含まれている式の集合関数の評価では、COUNT(*) を除き、「**集合関数では、NULL 値は無視されます。**」 (SQLSTATE=01003) という警告が生成されることがあります。ansinull を Off に設定した場合、この警告は表示されません。

制限事項

- ansinull を Off に設定した場合、影響を受けるのは、SELECT 文、UPDATEDELETE 文、INSERT 文の述部である WHERE、HAVING、ON だけです。CASE 文、IF 文、または IF 式の比較のセマンティックは、影響を受けません。
- Adaptive Server Enterprise 12.5 では、ansinull が Off に設定された場合、NULL パターン文字列を持つ LIKE 述部の動作が変更されています。SQL Anywhere では、LIKE 述部は従来どおり ansinull の設定に影響を受けません。

auditing オプション [データベース]

データベースでの監査を有効または無効にします。

指定可能な値

On、Off

デフォルト

Off

スコープ

PUBLIC グループのみに設定できます。すぐに有効になります。このオプションを設定するには、DBA パーミッションが必要です。

備考

このオプションは、監査のオンとオフを切り替えます。

監査とは、データベース内の多数のイベントに関する詳しい情報をトランザクション・ログに記録することをいいます。監査を行うと、パフォーマンスに少し負荷がかかりますが、いくつかのセキュリティ機能を得られます。データベースの監査をオンにした場合、トランザクション・ロ

グの使用を停止することはできません。トランザクション・ログを停止するためには、監査をオフにする必要があります。監査がオンであるデータベースを読み込み専用モードで起動することはできません。

auditing オプションを機能させるには、auditing オプションを On に設定し、sa_enable_auditing_type システム・プロシージャを使用して監査の対象とする情報のタイプを指定する必要があります。監査は、次のいずれかが該当する場合、有効になりません。

- auditing オプションが Off に設定されている
- 監査オプションが無効になっている

auditing オプションを On に設定し、監査オプションを指定しない場合は、すべてのタイプの監査情報が記録されます。または、パーミッションのチェック、接続試行、DDL 文、public オプション、トリガの中から組み合わせを選択して記録できます。

参照

- 「データベース・アクティビティの監査」 930 ページ
- 「sa_enable_auditing_type システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
- 「sa_disable_auditing_type システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』

例

監査をオンにします。

```
SET OPTION PUBLIC.auditing = 'On';
```

auditing_options オプション [データベース]

このオプションはシステムで使用するために予約されています。このオプションの設定は変更しないでください。

automatic_timestamp オプション [互換性]

TIMESTAMP データ型での新規カラムの解釈を制御します。

指定可能な値

On、Off

デフォルト

Off

備考

TIMESTAMP データ型の新規カラムに明示的なデフォルト値が定義されていない場合、デフォルトとして Transact-SQL の timestamp 値を適用するかどうかを制御します。automatic_timestamp オプションは Transact-SQL との互換性を保つために実装されています。

参照

- 「Transact-SQL との互換性を維持するためのオプション設定」 『SQL Anywhere サーバ - SQL の使用法』

background_priority オプション [データベース]

現在の接続以外の接続のパフォーマンスに対する影響を制限します。

指定可能な値

On、Off

デフォルト

Off

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

このオプションを一時的に設定した場合、この設定は現在の接続にのみ適用されます。同じユーザ ID による別の接続では、このオプションの設定を変えることができます。

background_priority がオンに設定されている場合、クエリ内並列処理は接続に使用されません。「クエリ実行時の並列処理」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

備考

On に設定すると、現在の接続が他の接続においてのパフォーマンスに対して最小限の影響しか及ぼさないように要求します。このオプションを使用すると、応答性が重要なタスクを、パフォーマンスがそれほど重要でない他のタスクと共存させることができます。

blob_threshold オプション [SQL Remote]

Message Agent がロング・オブジェクト (BLOB) として処理する値のサイズを制御します。

指定可能な値

整数 (バイト)

デフォルト

256

備考

blob_threshold オプションより長い値は、BLOB としてレプリケートされます。つまり、細かく分割して各部分をレプリケートしてから、SQL 変数を使用し、受信者サイトでその分割された部分を連結して再構成します。

SQL 文は、それぞれ 1 つのメッセージ内に収まる必要があります。したがって、このオプションの値をメッセージのサイズ (デフォルトでは 50 KB) より大きい値に設定しないでください。

参照

- 「SQL Remote オプション」 『SQL Remote』

blocking オプション [データベース]

ロック競合に応じる動作を制御します。

指定可能な値

On、Off

デフォルト

On

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

blocking オプションを On に設定すると、あるトランザクションが取得しようとしたロックが既存のロックと競合している場合、競合するすべてのロックが解放されるか、blocking_timeout に設定されたタイムアウト時間が経過するまで、そのトランザクションは待機状態となります。blocking_timeout ミリ秒以内にロックが解放されない場合は、待機しているトランザクションにエラーが返されます。blocking を Off に設定した場合は、トランザクションが競合するロックを取得しようとすると、エラーが返されます。

参照

- 「blocking_timeout オプション [データベース]」 431 ページ

blocking_timeout オプション [データベース]

トランザクションがロックを獲得するまで、どの程度の期間待機するかを制御します。

指定可能な値

整数 (ミリ秒)

デフォルト

0

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

blocking オプションを On に設定すると、 トランザクションが取得しようとしたロックが既存のロックと競合している場合は、blocking_timeout に設定された時間 (ミリ秒単位) が経過するま

で、競合するロックが解放されるのを待ちます。blocking_timeout ミリ秒以内にロックが解放されない場合は、待機しているトランザクションにエラーが返されます。

このオプションを 0 に設定すると、ロックを獲得しようとしているすべてのトランザクションは、競合するトランザクションがロックを解放するまで待機します。

参照

- 「blocking オプション [データベース]」 431 ページ

chained オプション [互換性]

BEGIN TRANSACTION 文がない場合のトランザクション・モードを制御します。

指定可能な値

On、Off

デフォルト

On

Open Client 接続と jConnect 接続の場合は Off

備考

Transact-SQL トランザクション・モードを制御します。非連鎖モード (chained = Off) では、明示的な BEGIN TRANSACTION 文が実行されてトランザクションを開始しないかぎり、各文が個々にコミットされます。連鎖モード (chained = On) では、データ検索文または修正文の前にトランザクションが暗黙的に開始されます。

checkpoint_time オプション [データベース]

データベース・サーバが、あるチェックポイントを実行してから次のチェックポイントを実行するまでの最大時間 (分単位) を設定します。

指定可能な値

整数

デフォルト

60

スコープ

PUBLIC グループのみに設定できます。このオプションを設定するには、DBA 権限が必要です。変更を有効にするには、データベース・サーバを停止し、再起動します。

備考

このオプションは、いつチェックポイントを実行すべきかを決定するために、recovery_time オプションと一緒に使用します。

参照

- 「チェックポイントとチェックポイント・ログ」 835 ページ
- 「recovery_time オプション [データベース]」 488 ページ

cis_option オプション [データベース]

リモート・データ・アクセス用のデバッグ情報をサーバ・メッセージ・ウィンドウに表示するかどうかを制御します。

指定可能な値

0, 7

デフォルト

0

スコープ

個々の接続または PUBLIC グループに設定できます。

備考

このオプションは、リモート・データ・アクセスを使用する場合に、リモート・データベースでのクエリの実行方法に関する情報をサーバ・メッセージ・ウィンドウに表示するかどうかを制御します。このオプションを 7 に設定すると、デバッグ情報がサーバ・メッセージ・ウィンドウに表示されます。このオプションを 0 (デフォルト) に設定すると、リモート・データ・アクセスのデバッグ情報はサーバ・メッセージ・ウィンドウには表示されません。

リモート・トレーシングを有効にすると、トレーシング情報がサーバ・メッセージ・ウィンドウに表示されます。この出力をファイルに記録するには、データベース・サーバの起動時に -o サーバ・オプションを指定します。「-o サーバ・オプション」 182 ページを参照してください。

cis_rowset_size オプション [データベース]

各フェッチに対してリモート・サーバから返されるローの数を設定します。

指定可能な値

整数

デフォルト

50

スコープ

個々の接続または PUBLIC グループに設定できます。リモート・サーバへの新しい接続が確立されたときに有効になります。

備考

このオプションは、ODBC を使用してリモート・データベース・サーバに接続する場合の ODBC FetchArraySize 値を設定します。

close_on_endtrans オプション [互換性]

トランザクションの終了時にカーソルを閉じるかどうかを制御します。

指定可能な値

On、Off

デフォルト

On

jConnect 接続の場合は Off

備考

close_on_endtrans が On に設定されている場合、カーソルが WITH HOLD で開かれていないかぎり、カーソルはトランザクションがコミットされるたびに閉じます。トランザクションがロールバックするときの動作は ansi_close_cursors_on_rollback オプションの設定内容によって制御されます。

close_on_endtrans が Off に設定されている場合は、ansi_close_cursors_on_rollback オプションの設定に関係なく、またカーソルが WITH HOLD で開かれているかどうかにも関係なく、コミットまたはロールバックのどちらでもカーソルは閉じません。

このオプションを Off に設定すると、Adaptive Server Enterprise と互換性のある動作が得られます。

参照

- 「ansi_close_cursors_on_rollback オプション [互換性]」 424 ページ

collect_statistics_on_dml_updates オプション [データベース]

INSERT、DELETE、UPDATE などのデータ変更 DML 文の実行中に統計を収集するかどうかを制御します。

指定可能な値

On、Off

デフォルト

On

備考

サーバは通常の文の実行中に統計情報を更新し、収集した統計を使用してカラム統計の自己チューニングを行います。collect_statistics_on_dml_updates オプションを Off に設定すると、INSERT、DELETE、UPDATE などのデータ変更 DML 文の実行中における統計の更新が無効となります。

通常の環境において、このオプションをオフにする必要はありません。ただし、きわめて大量のデータが頻繁に変更される環境では、このオプションを Off に設定することにより、パフォーマンスが向上することがあります。ただし、update_statistics も On に設定することが必要です。

collect_statistics_on_dml_updates オプションと update_statistics オプションが異なる点は、update_statistics オプションを On に設定した場合、ある述部を満たすローの実際の数とその述部を満たすと予想されるローの数が比較され、その結果に基づいて推定値が更新されることです。collect_statistics_on_dml_updates オプションを On に設定した場合は、挿入、更新、または削除されたローの値に基づいてカラム統計が修正されます。

参照

- 「update_statistics オプション [データベース]」 510 ページ
- 「カラム統計の更新」 『SQL Anywhere サーバ - SQL の使用法』

compression オプション [SQL Remote]

SQL Remote メッセージの圧縮レベルを設定します。

指定可能な値

-1 ～ 9 の整数

デフォルト

6

備考

値には次の意味があります。

- **-1** メッセージをバージョン 5 フォーマットで送信します。バージョン 5 の Message Agent では、バージョン 6 以降の Message Agent で送信されたメッセージを読むことはできません。使用しているシステムのすべての Message Agent をバージョン 6 以降にアップグレードするまでは、compression オプションを -1 に設定してください。
- **0** 圧縮しません。
- **1 ～ 9** 圧縮率を高めます。大きい圧縮率でメッセージを作成すると、小さい圧縮率を使用した場合より時間がかかります。

参照

- 「SQL Remote オプション」 『SQL Remote』

conn_auditing オプション [データベース]

auditing オプションが On に設定されている場合に、接続ごとに監査を有効または無効にします。

指定可能な値

On、Off

デフォルト

On

スコープ

現在の接続の間、テンポラリ・オプションとしてのみ設定できます。このオプションを設定するには、DBA パーミッションが必要です。

備考

conn_auditing オプションの設定は、ログイン・プロシージャに設定されている (login_procedure データベース・オプションで指定します) 場合のみ有効です。conn_auditing を On に設定すると、その設定の監査がオンになります。ただし、auditing オプションも On に設定しないかぎり、監査情報は記録されません。接続が監査されているかどうかを確認するには、次のプロシージャを実行します。

```
SELECT CONNECTION_PROPERTY ( 'conn_auditing' );
```

参照

- 「監査の有効化」 930 ページ
- 「auditing オプション [データベース]」 428 ページ
- 「login_procedure オプション [データベース]」 460 ページ

connection_authentication オプション [データベース]

認証アプリケーションのデータベース・シグニチャに対する、アプリケーション・シグニチャの確認に使用する認証文字列を指定します。

指定可能な値

文字列

デフォルト

空の文字列

スコープ

個々の接続に対してのみ設定できます。

備考

このオプションは、SQL Anywhere データベース・サーバの OEM 版を使用している場合にのみ有効になります。

認証アプリケーションは、接続確立後すぐに、すべての接続に対して connection_authentication データベース・オプションを設定する必要があります。シグニチャが確認されると、接続が認証

され、SQL パーミッションで設定されている内容にかかわらず、アクティビティは制限されません。シグニチャが確認されないと、接続は、非認証アプリケーションによって許可されているアクションに制限されます。

connection_authentication オプションは、TEMPORARY キーワードを使用して、現在の接続の間だけ設定できます。次の SQL 文は、接続を認証します。

```
SET TEMPORARY OPTION connection_authentication =
    'company = company-name;
    application = application-name;
    signature = application-signature'
```

company-name と *application-name* は、データベース認証の文と一致する必要があります。*application-signature* は、Sybase から取得するアプリケーション・シグニチャです。

会社名に引用符やアポストロフィなどの特殊文字が含まれている場合、受け入れられるようにするには、該当する文字を 2 つ続けて指定します。

SQL Anywhere の OEM 版の構成と使用の詳細については、「SQL Anywhere の認証アプリケーションの実行」 48 ページを参照してください。

参照

- 「database_authentication [データベース]」 440 ページ

例

次の例では、特殊文字を含む認証文字列を指定します。

```
SET TEMPORARY OPTION connection_authentication=
  'Company = Joe''s Garage;
   Application = Joe''s Program;
   Signature = 0fa55157edb8e14d818e...'
```

continue_after_raiserror オプション [互換性]

RAISERROR 文に応じて動作を制御します。

指定可能な値

On、Off

デフォルト

On

備考

RAISERROR 文はプロシージャ中で使用され、エラー生成を実行します。このオプションを Off に設定すると、RAISERROR 文に到達するたびにプロシージャまたはトリガの実行が停止します。

continue_after_raiserror オプションを On に設定すると、RAISERROR 文は実行終了エラーの信号を送信しなくなります。代わりに、RAISERROR ステータス・コードとメッセージが格納され、プロシージャが完了すると直前の RAISERROR が返されます。RAISERROR を起こしたプロシー

ジャが別のプロシージャに呼び出されると、RAISERROR は最も外側のプロシージャが終了するまで戻りません。

中間レベルの RAISERROR のステータスとコードは、プロシージャが終了すると失われます。リターン時に RAISERROR と一緒にエラーが生じた場合は、新しいエラー情報が戻され、RAISERROR 情報は失われます。アプリケーションは異なる実行ポイントで @@error グローバル変数を検査して、中間 RAISERROR ステータスを問い合わせることができます。

continue_after_raiserror オプションの設定は RAISERROR 文の後の動作を制御することを目的とし、on_tsql_error オプションが Conditional (デフォルト) に設定されている場合のみ使用します。on_tsql_error オプションを Stop または Continue に設定した場合、その設定は continue_after_raiserror の設定より優先されます。

参照

- 「on_tsql_error オプション [互換性]」 475 ページ

conversion_error オプション [互換性]

データベースからデータをフェッチするときの、データ型変換障害のレポートを制御します。

指定可能な値

On、Off

デフォルト

On

備考

このオプションは、データがデータベースからフェッチされるとき、またはデータベースに挿入されるときにデータ型変換障害が発生した場合、データベースがエラー (conversion_error が On) と警告 (conversion_error が Off) のどちらとしてレポートするかを制御します。

conversion_erro を On に設定すると、SQLE_CONVERSION_ERROR エラーが生成されます。このオプションを Off に設定すると、警告 SQLE_CANNOT_CONVERT が生成されます。

変換エラーが警告のみでレポートされた場合、変換できなかった値の代わりに NULL 値が使用されます。Embedded SQL では、インジケータ変数はエラーを出したカラムに -2 を設定します。

cooperative_commit_timeout オプション [データベース]

トランザクション・ログ内の COMMIT エントリがいつディスクに書き込まれるかを管理します。

指定可能な値

整数 (ミリ秒)

デフォルト

250

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

このオプションは、cooperative_commits が On に設定されている場合にかぎり意味を持ちます。データベース・サーバは、ディスクに書き込む前に、指定されたミリ秒数だけ、他の接続がログのページを埋めるのを待ちます。デフォルト設定は 250 ミリ秒です。

参照

- 「cooperative_commits オプション [データベース]」 439 ページ

cooperative_commits オプション [データベース]

コミットがいつディスクに書き込まれるかを制御します。

指定可能な値

On、Off

デフォルト

On

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

cooperative_commits を Off に設定した場合、COMMIT はデータベース・サーバで受信されるとすぐにディスクに書き込まれ、その後アプリケーションは継続が許可されます。

cooperative_commits を On (デフォルト) に設定し、他にアクティブな接続がある場合は、データベース・サーバは COMMIT をすぐにはディスクに書き込みません。アプリケーションは cooperative_commit_timeout オプションで設定した最大長になるまで、他にそのページに含めるものがないか待ってから、COMMIT をディスクに書き込みます。

cooperative_commits を On に設定し、cooperative_commit_timeout の設定を大きくすると、ディスク I/O の数が減少することによってデータベース・サーバの全体的なスループットが向上しますが、各接続のターンアラウンド・タイムが長くなります。

cooperative_commits と delayed_commits の両方を On に設定している場合、cooperative_commit_timeout に設定された時間がページの書き込みなしで経過すると、アプリケーションが (コミットが実行されたかのように) 再開され、残りの時間 (delayed_commit_timeout の値から cooperative_commit_timeout の値を差し引いた長さ) は delayed_commits 間隔として使用されます。その後、ページは満杯になっていなくてもディスクに書き込まれます。

参照

- 「delayed_commits オプション [データベース]」 446 ページ

database_authentication [データベース]

データベースびの認証文字列を設定します。

指定可能な値

文字列

デフォルト

空の文字列

スコープ

PUBLIC グループのみに設定できます。このオプションを有効にするには、データベースを再起動する必要があります。

備考

このオプションは、SQL Anywhere データベース・サーバの OEM 版を使用している場合にのみ有効になります。

データベースが認証されると、正しい認証シグネチャを指定する接続だけが、データベースへの操作を実行できます。認証されない接続は、読み込み専用モードで操作します。認証データベースを使用する場合は、SQL Anywhere の OEM 版を使用する必要があります。

データベースを認証するには、データベースの database_authentication オプションを設定します。

```
SET OPTION PUBLIC.database_authentication =
   'company = company-name;
   application = application-name;
   signature = database-signature'
```

company-name と *application-name* の引数は、シグニチャの取得時に Sybase に提供した値です。*database-signature* は、Sybase から受け取ったデータベース・シグニチャです。

会社名に引用符やアポストロフィなどの特殊文字が含まれている場合、受け入れられるようにするには、該当する文字を 2 つ続けて指定します。

データベース・サーバが認証データベースをロードするとき、サーバ・メッセージのウィンドウに、認証された会社とアプリケーションについて説明するメッセージが表示されます。このメッセージを調べて、database_authentication オプションが有効になっているかどうかを確認できます。メッセージの形式は次のとおりです。

```
This database is licensed for use with:
Application: application-name
Company: company-name
```

SQL スクリプト・ファイルに認証の文を格納し、長いシグニチャを繰り返し入力することを回避できます。認証の文をファイル *install-dir¥scripts¥authenticate.sql* に格納すると、この文は、データベースの作成、再構築、または更新を行うたびに適用されます。

SQL Anywhere の OEM 版の構成と使用の詳細については、「SQL Anywhere の認証アプリケーションの実行」 48 ページを参照してください。

参照

- 「connection_authentication オプション [データベース]」 436 ページ

例

```
SET OPTION PUBLIC.database_authentication =
    'company = company-name;
    application = application-name;
    signature = database-signature'
```

date_format オプション [互換性]

データベースから取り出した日付のフォーマットを設定します。

日付フォーマットの解釈の制御については、「date_order オプション [互換性]」 443 ページを参照してください。

指定可能な値

文字列

デフォルト

'YYYY-MM-DD' (ISO 日付フォーマット仕様に準拠)

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

フォーマットは次の記号を組み合わせた文字列です。

シンボル	説明
yy	2 桁の年
yyyy	4 桁の年
mm	2 桁の月
mmm[*m*…]	月を示す略式文字
d	曜日を示す 1 桁の数字 (0 = 日曜、6 = 土曜)
dd	2 桁の日
ddd[*d*…]	曜日を示す略式文字
jjj	通し日数 (1 ー 366)

各記号は、フォーマットされる日付のデータで置き換えられます。

文字データがマルチバイトである場合、各記号の長さはバイト数ではなく文字数を表します。たとえば、'mmm' は 3 文字の月名を示しています。

文字データを表す記号 (*mmm* など) では、出力の文字を次のように制御できます。

- 記号をすべて大文字で入力すると、フォーマットはすべて大文字で表記されます。たとえば、MMM と入力すると、JAN と表記されます。
- 記号をすべて小文字で入力すると、フォーマットはすべて小文字で表記されます。たとえば、mmm と入力すると、jan と表記されます。
- 大文字と小文字を混ぜて入力すると、使用される言語に適切な文字が SQL Anywhere により選択されます。たとえば、Mmm と入力すると、英語では May、フランス語では Mai と表記されます。

数値データを表す記号では、記号に大文字を使用するか小文字を使用するかで 0 埋め込みを制御できます。

- 記号をすべて大文字または小文字 (MM や mm など) で入力すると、0 埋め込みが行われます。たとえば、yyyy/mm/dd と入力すると、2002/01/01 と表記されます。
- 大文字と小文字を混ぜて入力すると (Mm など)、0 の埋め込みは行われません。たとえば、yyyy/Mm/Dd と入力すると、2002/1/1 と表記されます。

参照

- 「time_format オプション [互換性]」 504 ページ
- 「timestamp_format オプション [互換性]」 505 ページ

例

次の表は、2006 年 6 月 2 日 (金) に実行された次の文からの出力と、date_format の設定を示します。

```
SELECT CURRENT DATE
```

date_format	SELECT CURRENT DATE
yyyy/mm/dd/ddd	02.06.06/fri
yyyy/Mm/Dd/ddd	2006/6/2/fri
jjj	153
mmm yyyy	jun 2006
Mmm yyyy	Jun 2006
mm-yyyy	06-2006

date_order オプション [互換性]

日付フォーマットの解釈を制御します。

データベースから取得した日付のフォーマット設定の詳細については、「date_format オプション [互換性]」 441 ページを参照してください。

指定可能な値

MDY、YMD、DMY

デフォルト

YMD (ISO 日付フォーマット仕様に準拠)

Open Client 接続と jConnect 接続の場合、デフォルトは MDY に設定されます。

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

データベース・オプション date_order では、10/11/12 が Oct 11 1912、Nov 12 1910、Nov 10 1912 のうち、どの日付を意味するかを指定します。

debug_messages オプション [データベース]

DEBUG ONLY 句を含む MESSAGE 文を実行するか否かを制御します。

指定可能な値

On、Off

デフォルト

Off

備考

このオプションにより、指定した DEBUG ONLY 句がある MESSAGE 文を含むストアド・プロシージャとトリガ内のデバッグ・メッセージの動作を制御できます。このオプションはデフォルトで Off に設定されており、MESSAGE 文が実行された場合にデバッグ・メッセージは表示されません。debug_messages を On に設定することにより、すべてのストアド・プロシージャとトリガでデバッグ・メッセージが有効になります。

> **注意**
> debug_messages オプションを Off に設定している場合、DEBUG ONLY メッセージはパフォーマンスへの影響が小さいため、通常は運用システム上のストアド・プロシージャに残すことができます。ただし、このオプションを頻繁に使用する場合は慎重に使用してください。パフォーマンスが多少低下する場合があります。

参照

- 「MESSAGE 文」 『SQL Anywhere サーバ - SQL リファレンス』

dedicated_task オプション [データベース]

要求処理タスクを、単一の接続からの要求処理に特化します。

指定可能な値

On、Off

デフォルト

Off

スコープ

現在の接続の間、テンポラリ・オプションとしてのみ設定できます。このオプションを設定するには、DBA パーミッションが必要です。

備考

dedicated_task 接続オプションを On に設定すると、要求処理タスクは、その接続の要求処理専用となります。このオプションを有効にして接続を事前に確立することで、応答しなくならないかぎりデータベース・サーバのステータスに関する情報を集められます。

default_dbspace オプション [データベース]

テーブルが作成されるデフォルト DB 領域を変更します。

指定可能な値

文字列

デフォルト

空の文字列

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

各データベースについて、システム (メイン) DB 領域に加えて最大 12 の DB 領域を作成できます。DB 領域を指定しないでテーブルを作成すると、このオプションに指定された DB 領域が使用されます。このオプションを設定しない場合、空の文字列に設定した場合、または system に設定した場合は、システム DB 領域が使用されます。

すべてのテーブルをシステム DB 領域以外のロケーションに作成している場合は、システム DB 領域はチェックポイント・ログとシステム・テーブルの保存にのみ使用されます。この設定は、パフォーマンス上の理由からチェックポイント・ログを他のデータベース・オブジェクトとは別のディスクに保存する場合に便利です。チェックポイント・ログを別のディスクに保存するに

は、すべての CREATE TABLE 文で DB 領域を指定するか、このオプションの設定を変更してからテーブルを作成します。

参照

- 「異なるファイルの異なるデバイスへの配置」『SQL Anywhere サーバ - SQL の使用法』
- 「CREATE DBSPACE 文」『SQL Anywhere サーバ - SQL リファレンス』

例

次の例では、まず library という名前の新しい DB 領域を作成します。次に、その DB 領域をデフォルト DB 領域に設定し、テーブル LibraryBooks をシステム DB 領域ではなく library DB 領域に保存します。

```
CREATE DBSPACE library
AS 'e:¥¥dbfiles¥¥library.db';
SET OPTION default_dbspace = 'library';
CREATE TABLE LibraryBooks (
  title CHAR(100),
  author CHAR(50),
  isbn CHAR(30),
);
```

default_timestamp_increment オプション [データベース] [Mobile Link クライアント]

カラム中の値をユニークにするために TIMESTAMP データ型のカラムに追加する時間 (マイクロ秒単位) を指定します。

指定可能な値

整数 (1 ～ 1000000)

デフォルト

1

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

SQL Anywhere では、TIMESTAMP 値の精度が小数第 6 位であるため、2 つの同一の TIMESTAMP 値を区別するために、デフォルトでは 1 マイクロ秒 (0.000001 秒) が追加されます。

Microsoft Access などのソフトウェアの中には、TIMESTAMP 値を小数点第 3 位までトランケートしてしまうため、正しい比較を行う上で問題になるものもあります。互換性を確保するために、truncate_timestamp_values オプションを On に設定し、SQL Anywhere が格納する小数点以下の桁数を指定できます。

Mobile Link 同期に対し、このオプションを設定する場合は、最初の同期の実行前に設定する必要があります。

参照

- 「truncate_timestamp_values オプション [データベース] [Mobile Link クライアント]」 507 ページ

delayed_commit_timeout オプション [データベース]

delayed_commits オプションが On のときに、アプリケーションが COMMIT を実行する時間から、COMMIT が実際にディスクに書き込まれる時間までの最大遅延を指定します。

指定可能な値

整数 (ミリ秒)

デフォルト

500

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

このオプションは、delayed_commits が On に設定されているときにのみ有効になります。このオプションは、トランザクション・ログの COMMIT エントリをどの時点でディスクに書き込むかを管理します。delayed_commits を On に設定すると、データベース・サーバは、delayed_commit_timeout オプションで指定されたミリ秒間待ち、他の接続によって 1 つのログ・ページが満杯になってから、現在のページ内容をディスクに書き込みます。「delayed_commits オプション [データベース]」 446 ページを参照してください。

delayed_commits オプション [データベース]

COMMIT の後でサーバが制御をアプリケーションにいつ戻すかを決定します。

指定可能な値

On、Off

デフォルト

Off (ISO の COMMIT 動作に準拠)

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

On に設定すると、データベース・サーバは COMMIT のトランザクション・ログ・エントリがディスクに書き込まれるのを待たずに、直ちに COMMIT 文に応答します。Off に設定すると、アプリケーションは COMMIT がディスクに書き込まれるまで待たなければなりません。

このオプションが On の場合、ログ・ページが満杯となったときと、delayed_commit_timeout オプションに設定された時間が経過したときのうち、いずれか早い時点でログがディスクに書き込まれます。サーバが COMMIT に応答した後で、ページがディスクに書き込まれる前にシステム障害が起こった場合に、コミットされた場合にでもトランザクションが失われる可能性がわずかにあります。delayed_commits を On に設定し、delayed_commit_timeout オプションを高い値にすると、応答時間は早くなりますが、リカバリ中にコミットされたトランザクションが失われるリスクが若干高くなります。

cooperative_commits と delayed_commits の両方を On に設定している場合、cooperative_commit_timeout オプションに設定された時間がページへの書き込みなしで経過すると、アプリケーションが (コミットが実行されたかのように) 再開されます。残りの時間 (delayed_commit_timeout の値から cooperative_commit_timeout の値を差し引いた長さ) は delayed_commits 間隔として使用され、その時間が経過すると、ページは満杯になっていなくてもディスクに書き込まれます。

参照

- 「cooperative_commit_timeout オプション [データベース]」 438 ページ
- 「cooperative_commits オプション [データベース]」 439 ページ
- 「delayed_commit_timeout オプション [データベース]」 446 ページ

delete_old_logs オプション [Mobile Link クライアント] [SQL Remote] [Replication Agent]

トランザクションがレプリケートまたは同期されたときにトランザクション・ログを削除するかどうかを制御します。

指定可能な値

On、Off、Delay、*n* days

デフォルト

Off

備考

このオプションは、SQL Anywhere Mobile Link クライアント、SQL Remote、SQL Anywhere Replication Agent によって使用されます。デフォルト設定は Off です。このオプションを On に設定すると、古いトランザクション・ログ内のすべての変更についてレプリケーションまたは同期が完了した時点で、古いトランザクション・ログが削除されます。Delay に設定すると、古いトランザクション・ログのファイル名に作成日が現在の日付であることが示されている場合、すべての変更が送信され、受信が確認されても、そのトランザクション・ログは削除されません。*n* days に設定すると、*n* days より古いログが削除されます。

delete_old_logs オプションを BACKUP 文と一緒に使用して古いトランザクション・ログのコピーを削除する方法については、「BACKUP 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

例

delete_old_logs オプションが 10 days に設定されているリモート・データベースに対して、1 月 18 日に dbmlsync を実行すると、dbmlsync は、1 月 7 日以前に作成されたオフライン・トランザクション・ログを削除します。リモート・データベースは、オプションを次のように設定します。

```
SET OPTION delete_old_logs = '10 days'
```

参照

- 「SQLAnywhere クライアントのロギング」 『Mobile Link - クライアント管理』
- 「MirrorLogDirectory (mld) 拡張オプション」 『Mobile Link - クライアント管理』
- 「SQL Remote オプション」 『SQL Remote』

divide_by_zero_error オプション [互換性]

ゼロ除算のレポートを制御します。

指定可能な値

On、Off

デフォルト

On

備考

このオプションは、ゼロ除算をエラーとしてレポートするかどうかを指示します。このオプションを On に設定すると、ゼロ除算は SQLSTATE 22012 のエラーになります。

このオプションを Off に設定すると、ゼロ除算はエラーになりません。代わりに、NULL が返されます。

encrypt_aes_random_iv オプション [データベース]

暗号化の出力が決定的であるかどうかを制御します。

指定可能な値

On、Off

デフォルト

On

備考

On に設定すると、同一の暗号化キーとともに同一の入力文字列を入力したときに、暗号化の出力 (暗号文) が異なります。Off に設定すると、同一の入力文字列と暗号化キーを入力したときに、暗号化の出力が同一になります。

参照

- 「トランスポート・レイヤ・セキュリティ」 949 ページ
- 「ENCRYPT 関数 [文字列]」 『SQL Anywhere サーバ - SQL リファレンス』

escape_character オプション [互換性]

このオプションはシステムで使用するために予約されています。このオプションの設定は変更しないでください。

exclude_operators オプション [データベース]

このオプションはシステムで使用するために予約されています。このオプションの設定は変更しないでください。

extended_join_syntax オプション [データベース]

複数テーブルのジョインに対して、重複する相関名構文を持つクエリを許可するか、またはエラーとしてレポートするかを指定します。

指定可能な値

On、Off

デフォルト

On

備考

このオプションを On に設定すると、外部ジョインの NULL 入力側で重複した相関名を使用することが可能となります。同じ相関名で指定されたすべてのテーブルまたはビューは、テーブルまたはビューの同じインスタンスとして解釈されます。

次の FROM 句は、重複する相関名を使用したジョインが SQL Anywhere でどのように解釈されるかを示しています。C1 と C2 は検索条件を表します。

```
( R left outer join T on ( C1 ), T  join S on ( C2 ) )
```

このオプションを On に設定すると、このジョインは次のように解釈されます。

```
( R left outer join T on ( C1 ) ) join S on ( C2 )
```

このオプションを Off に設定すると、次のエラーが生成されます。

SQL Anywhere エラー -137: テーブル 'T' にはユニークな相関名が必要です。

注意
重複する相関名を削除した結果を確認する場合、2 番目の引数を ANSI に設定した REWRITE 関数を使うと、書き直した文を表示できます。

参照

- 「REWRITE 関数 [その他]」『SQL Anywhere サーバ - SQL リファレンス』

external_remote_options [SQL Remote]

メッセージ・リンク・パラメータを格納するかどうかを指定します。

指定可能な値

On、Off

デフォルト

Off

備考

このオプションは SQL Remote が、メッセージ・リンク・パラメータをデータベースに格納するか (Off)、外部に格納するか (On) を指定するために使用します。

fire_triggers オプション [互換性]

データベースにおけるトリガ起動を制御します。

指定可能な値

On、Off

デフォルト

On

備考

On に設定すると、トリガが起動されます。Off に設定すると、参照整合性トリガ (カスケード更新や削除など) を含め、トリガは起動されません。DBA 権限を持つユーザだけがこのオプションを設定できます。このオプションは -gf オプションによって上書きされます。このオプションを指定した場合、fire_triggers の設定にかかわらず、すべてのトリガ起動がオフになります。

Adaptive Server Enterprise トランザクション・ログのアクションは、トリガによって実行されるアクションも含め、すべて SQL Anywhere にレプリケートされるので、このオプションは、Adaptive Server Enterprise から SQL Anywhere にデータをレプリケートする場合に意味を持ちます。

first_day_of_week オプション [データベース]

何曜日を週の最初にするかを設定します。

指定可能な値

1, 2, 3, 4, 5, 6, 7

デフォルト

7 (1 週間は日曜日から始まる)

備考

値には次の意味があります。

値	意味
1	月曜日
2	火曜日
3	水曜日
4	木曜日
5	金曜日
6	土曜日
7	日曜日

このオプションで指定する値は、平日の値を取得するときに DATEPART 関数の結果に影響します。週の最初の曜日は、SET 文の DATEFIRST オプションを使用して変更することもできます。

このオプションで指定する値は、DOW 関数の結果には影響しません。たとえば、週の最初の曜日を月曜日に設定しても、DOW 関数は月曜日の値として 2 を返します。

参照

- 「DATEPART 関数 [日付と時刻]」『SQL Anywhere サーバ - SQL リファレンス』
- 「SET 文 [T-SQL]」『SQL Anywhere サーバ - SQL リファレンス』

float_as_double オプション [互換性]

FLOAT キーワードの解釈を制御します。

指定可能な値

On、Off

デフォルト

Off

Open Client 接続と JDBC 接続の場合は On

備考

精度が指定されていない場合、float_as_double オプションは、FLOAT キーワードが Adaptive Server Enterprise の FLOAT キーワードと同様の動作をするようにします。

使用可能な場合 (On のとき)、キーワード FLOAT のすべてのオカレンスは SQL 文内のキーワード DOUBLE と同等であると解釈されます。

デフォルトでは、SQL Anywhere の FLOAT 値は Adaptive Server Enterprise で REAL 値として解釈されます。Adaptive Server Enterprise は独自の FLOAT 値を DOUBLE として処理するので、このオプションを使用可能にすると、Enterprise が FLOAT 値を処理するのと同じ方法で SQL Anywhere は FLOAT 値を処理します。

REAL 値は 4 バイトで、DOUBLE 値は 8 バイトです。ANSI SQL/2003 仕様に応じて、FLOAT はプラットフォームに基づいて解釈できます。必要な精度を処理できるかぎり、値のサイズを決定するのはデータベースです。Adaptive Server Enterprise と SQL Anywhere は異なるデフォルト動作を示します。

float_as_double オプションは、精度が指定されていないときにだけ効力を持ちます。たとえば、次の文はオプション設定に影響されません。

```
CREATE TABLE t1(
  c1 float( 5 ) );
```

次の文は、オプション設定に影響されます。

```
CREATE TABLE t2(
  c1 float);
// affected by option setting
```

for_xml_null_treatment オプション [データベース]

FOR XML 句を使用するクエリでの NULL 値の処理を制御します。

指定可能な値

Empty、Omit

デフォルト

Omit

備考

FOR XML 句を含むクエリを実行する場合、for_xml_null_treatment オプションが NULL 値の処理方法を決定します。デフォルトで、NULL 値を含む要素と属性が結果から省略されます。このオプションを Empty に設定すると、値が NULL の場合に空の要素または属性が生成されます。

参照

- 「FOR XML 句を使用してクエリ結果を XML として取り出す」『SQL Anywhere サーバ - SQL の使用法』
- 「SELECT 文」『SQL Anywhere サーバ - SQL リファレンス』

force_view_creation オプション [データベース]

このオプションはシステムで使用するために予約されています。このオプションの設定は変更しないでください。

警告
force_view_creation オプションは、*reload.sql* スクリプトでのみ使用されます。このオプションは、アンロード・ユーティリティ (dbunload) でのみ使用され、明示的に設定されません。

global_database_id オプション [データベース]

DEFAULT GLOBAL AUTOINCREMENT が指定されたカラムの値の範囲を設定します。レプリケーション環境でユニークなプライマリ・キーを生成する場合に使用します。

指定可能な値

整数

デフォルト

2147483647

スコープ

PUBLIC グループのみに設定できます。DBA 権限が必要です。

備考

このオプションで指定した値は開始値となります。DEFAULT GLOBAL AUTOINCREMENT が指定されたカラムでは、DEFAULT GLOBAL AUTOINCREMENT カラムに値がないテーブルにローを挿入すると、そのカラムの値が自動的に生成されます。この値は、global_database_id の値とカラムの分割サイズによって決まります。

global_database_id をデフォルト値に設定すると、デフォルトのグローバル・オートインクリメントが無効になります。この場合、デフォルトとして NULL が生成されます。

現在のデータベースのオプション値を検索するには、次の文を使用します。

```
SELECT DB_PROPERTY( 'GlobalDBID' );
```

この機能は、特に、レプリケーション環境でユニークなプライマリ・キーを生成するために使用します。

参照

- 「CREATE TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「データベース・レベルのプロパティ」 550 ページ
- Mobile Link： 「グローバル・データベース ID の設定」『Mobile Link - サーバ管理』
- SQL Remote： 「ユニークなプライマリ・キーの確保」『SQL Remote』

例

次の例は、データベース ID 番号を 100 に設定します。

```
SET OPTION PUBLIC.global_database_id = '100';
```

http_session_timeout オプション [データベース]

クライアントが HTTP セッションがタイムアウトになったと判断するまで待機する時間 (分単位) を指定します。

指定可能な値

整数 (1 ー 525600)

デフォルト

30

スコープ

PUBLIC グループのみに設定できます。このオプションを設定するには、DBA 権限が必要です。

備考

このオプションは、Web サービス・アプリケーションに対してさまざまなセッション・タイムアウトを提供します。Web サービス・アプリケーションは、HTTP セッションを所有する任意の要求内でタイムアウトの値を変更できます。ただし、タイムアウト値を変更すると、HTTP セッションがタイムアウトになった場合に、それ以降にキューイングされた要求に影響します。Web アプリケーションには、存在しなくなった HTTP セッションへのアクセスをクライアントが試行しているかどうかを検出するロジックを含める必要があります。これは、SessionCreateTime 接続プロパティの値を調べて、タイムスタンプが有効であるかどうかを確認することで実行できます。対象の HTTP 要求が現在の HTTP セッションに関係ない場合は、SessionCreateTime 接続プロパティには空の文字列が設定されています。

参照

- SessionCreateTime プロパティと http_session_timeout プロパティ：「接続レベルのプロパティ」 518 ページ
- 「HTTP セッションの使用」 『SQL Anywhere サーバ - プログラミング』

integrated_server_name オプション [データベース]

統合化ログインのために Windows ユーザ・グループ・メンバシップを検索するために使用する Domain Controller サーバの名前を指定します。

指定可能な値

文字列

デフォルト

NULL

スコープ

PUBLIC グループのみに設定できます。このオプションを設定するには、DBA 権限が必要です。

備考

このオプションを使用すると、DBA は統合化ログインに Windows ユーザ・グループを使う場合に、グループ・メンバシップを検索するために使用する Domain Controller サーバの名前を指定で

きます。デフォルトでは、グループ・メンバシップの確認に SQL Anywhere データベース・サーバを実行中のコンピュータが使用されます。

参照

- 「Windows ユーザ・グループ用の統合化ログインの作成」 98 ページ
- 「GRANT 文」 『SQL Anywhere サーバ - SQL リファレンス』

例

次の例では、グループ・メンバシップがコンピュータ server-1 で検証されることを指定します。

```
SET OPTION PUBLIC.integrated_server_name = '¥¥server-1';
```

isolation_level オプション [互換性]

ロック独立性レベルを制御します。

指定可能な値

0、1、2、3、snapshot、statement-snapshot、readonly-statement-snapshot

デフォルト

0

Open Client 接続、jConnect 接続、TDS 接続の場合は 1

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

このオプションは、次のようにロック独立性レベルを制御します。

- **0** ダーティ・リード、繰り返し不可能読み出し、幻ローを許可します。
- **1** ダーティ・リードを防ぎます。繰り返し不可能読み出しと幻ローを許可します。
- **2** ダーティ・リードと繰り返し不可能読み出しを防ぎます。幻ローを許可します。
- **3** 直列化可能。ダーティ・リード、繰り返し不可能読み出し、幻ローを防ぎます。
- **snapshot** トランザクションが最初のローの読み込みまたは更新を行った時点から、コミットされたデータのスナップショットを使用します。
- **statement-snapshot** 個々の文について、データベースから最初のローが読み込まれた時点から、コミットされたデータのスナップショットを使用します。繰り返し不可能読み出しと幻ローは、1 つのトランザクション内で発生することはありますが、1 つの文の中で発生することはありません。
- **readonly-statement-snapshot** 読み込み専用の文についてのみ、データベースから最初のローが読み込まれた時点から、コミットされたデータのスナップショットを使用します。繰り返し不可能読み出しと幻ローは、1 つのトランザクション内で発生することはありますが、1

つの文の中で発生することはありません。更新可能な文については、updatable_statement_isolation オプションに指定された独立性レベル (0 (デフォルト)、1、2、3 のいずれか) を使用します。

「独立性レベルと一貫性」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

このオプションを snapshot、statement-snapshot、または readonly-statement-snapshot に設定する場合は、allow_snapshot_isolation を On に設定する必要があります。

iAnywhere JDBC ドライバを使用する場合は、デフォルトの独立性レベルは 0 となります。

snapshot、statement-snapshot、readonly-statement-snapshot のいずれかの独立性レベルでクエリを実行した場合には、データベースのコミットされた状態のスナップショットからデータが取得されます。

各文に OPTION 句を含めることによって、テンポラリ設定とパブリック設定よりも、それぞれの INSERT、UPDATE、DELETE、または SELECT 文におけるこのオプションの設定を優先することができます。次の項を参照してください。

- 「INSERT 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「UPDATE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「DELETE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「SELECT 文」『SQL Anywhere サーバ - SQL リファレンス』

参照

- 「allow_snapshot_isolation オプション [データベース]」 422 ページ
- 「updatable_statement_isolation オプション [データベース]」 510 ページ
- 「スナップショット・アイソレーション」『SQL Anywhere サーバ - SQL の使用法』
- 「独立性レベルと一貫性」『SQL Anywhere サーバ - SQL の使用法』
- 「独立性レベルの選択」『SQL Anywhere サーバ - SQL の使用法』

java_location オプション [データベース]

データベースの Java VM のパスを指定します。

指定可能な値

文字列

デフォルト

空の文字列

スコープ

PUBLIC グループのみに設定できます。このオプションを設定するには、DBA 権限が必要です。

備考

デフォルトでは、このオプションには空の文字列が設定されます。この場合、データベース・サーバは、JAVA_HOME 環境変数と Java VM のパスおよびその他のロケーションを検索しま

す。java_location を設定していない場合にデータベース・サーバが使用する Java VM を確認するには、JavaVM データベース・プロパティを使用します。

参照

- 「java_main_userid オプション [データベース]」 457 ページ
- 「java_vm_options オプション [データベース]」 457 ページ
- JavaVM プロパティ： 「データベース・レベルのプロパティ」 550 ページ
- 「Java VM の選択」 『SQL Anywhere サーバ - プログラミング』

java_main_userid オプション [データベース]

接続をクラスのインストールなどの Java 関連の管理タスクに使用できるデータベース・ユーザを指定します。

指定可能な値

文字列

デフォルト

DBA ユーザ (データベースの初期化中に作成されるデフォルト・ユーザ)

スコープ

PUBLIC グループのみに設定できます。DBA 権限が必要です。

備考

このオプションで指定するユーザ ID は、必要な操作を実行できるように DBA 権限を持っていなければなりません。このユーザのパスワードは必要ありません。

参照

- 「java_location オプション [データベース]」 456 ページ
- 「java_vm_options オプション [データベース]」 457 ページ
- 「Java VM の選択」 『SQL Anywhere サーバ - プログラミング』

java_vm_options オプション [データベース]

Java VM の起動時にデータベース・サーバが使用するコマンド・ライン・オプションを指定します。

指定可能な値

文字列

デフォルト

空の文字列

スコープ

PUBLIC グループのみに設定できます。DBA 権限が必要です。

備考

このオプションを使用すると、java_location オプションで指定された Java VM を起動するときにデータベース・サーバが使用するオプションを指定できます。これらの追加のオプションを使用して、デバッグ目的に Java VM を設定したり、UNIX プラットフォーム上のサービスとして実行したりできます。32 ビット・モードではなく 64 ビット・モードで Java VM を使用するために、その他のオプションが必要になることもあります。

参照

- 「java_location オプション [データベース]」 456 ページ
- 「java_main_userid オプション [データベース]」 457 ページ
- 「Java VM の選択」 『SQL Anywhere サーバ - プログラミング』

例

次の例では、java_vm_options オプションを使用することで、データベース・サーバがサービスとして起動され、ユーザがログアウトする必要が生じたときに、UNIX 上での Java VM の実行を維持しています。

```
SET OPTION PUBLIC.java_vm_options = '-Xrs'
```

次の例では、Java VM に対して HP-UX 上で 64 ビット・モードで実行するように指示します。

```
SET OPTION PUBLIC.java_vm_options = '-d64'
```

log_deadlocks オプション [データベース]

デッドロック・レポートのオン／オフを制御します。

指定可能な値

On、Off

デフォルト

Off

スコープ

PUBLIC グループのみに設定できます。DBA 権限が必要です。すぐに有効になります。

備考

このオプションを On に設定した場合、データベース・サーバは、内部バッファ内のデッドロックに関する情報をログに記録します。バッファのサイズは、10000 バイトに固定されています。デッドロック情報は sa_report_deadlocks ストアド・プロシージャを使用すると表示できます。このオプションを Off に設定すると、バッファの内容はクリアされます。

デッドロックが発生すると、そのデッドロックに関わった接続のみの情報がレポートされます。接続のレポート順序は、どの接続がどの行を待っているかに基づきます。スレッド・デッドロックの場合、すべての接続の情報がレポートされます。

参照

- 「sa_report_deadlocks システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
- 「ブロックされているユーザの判別」 『SQL Anywhere サーバ - SQL の使用法』

login_mode オプション [データベース]

データベースの統合化ログインと Kerberos ログインの使用を制御します。

指定可能な値

Standard、Integrated、Kerberos、Mixed (旧式) のうち、1 つまたは複数

デフォルト

Standard

スコープ

PUBLIC グループのみに設定できます。DBA 権限が必要です。すぐに有効になります。

備考

このオプションは、標準ログイン、統合化ログイン、Kerberos ログインを許可するかどうかを指定します。以下のログイン・モードのうち 1 つまたは複数を指定できます (大文字と小文字は区別されません)。

- **Standard** 標準ログインを許可します。これがデフォルト設定です。標準ログインでは、ユーザ ID とパスワードの両方を入力する必要があり、統合化接続や Kerberos 接続のパラメータは使用されません。
- **Integrated** 統合化ログインを許可します。
- **Kerberos** Kerberos ログインを許可します。
- **Mixed (旧式)** Standard,Integrated の指定と同じです。

複数のログイン・モードを指定すると、データベース・サーバでは、そのすべてのモードが使用可能となります。

> **警告**
> login_mode データベース・オプションを標準ログインが許可されない設定にした場合、接続できるのは、統合化ログイン・マッピングまたは Kerberos ログイン・マッピングを付与されているユーザまたはグループだけに制限されます。ユーザ ID とパスワードで接続しようとすると、エラーが発生します。唯一の例外は、DBA 権限を持つユーザです。

複数の値をカンマで区切ったリストとして指定できます。このリストに空白スペースを含めることはできません。たとえば、次の設定は標準ログインと統合化ログインの両方を許可します。

```
SET OPTION PUBLIC.login_mode = 'Standard,Integrated';
```

データベース・ファイルを保護していないため、権限のないユーザがコピーできる場合は、一時的なパブリック login_mode オプションを使用する必要があります (統合化ログインと Kerberos ログインの両方で)。ファイルをコピーする場合、統合化ログインと Kerberos ログインはデフォルトではサポートされません。

参照

- 「統合化ログインの使用方法」 98 ページ
- 「Kerberos 認証の使用」 108 ページ
- 「セキュリティについての考慮事項：コピーされたデータベース・ファイル」 119 ページ

例

次の文は、統合化ログインのみ有効にします (標準ログインと Kerberos ログインは失敗します)。

```
SET OPTION PUBLIC.login_mode = 'Integrated';
```

次の文は、標準ログインと Kerberos ログインを有効にします (統合化ログインは失敗します)。

```
SET OPTION PUBLIC.login_mode = 'Standard,Kerberos';
```

次の文は、標準ログイン、統合化ログイン、Kerberos ログインを有効にします。

```
SET OPTION PUBLIC.login_mode = 'Standard,Integrated,Kerberos';
```

login_procedure オプション [データベース]

起動時の接続互換性オプションを設定するログイン・プロシージャを指定します。デフォルトでは、このプロシージャは sp_login_environment プロシージャを呼び出して、どのオプションを設定するかを決定します。

指定可能な値

文字列

デフォルト

sp_login_environment

スコープ

DBA 権限が必要です。

備考

このログイン・プロシージャは、ランタイムに sp_login_environment プロシージャを呼び出して、データベース接続設定を決定します。このログイン・プロシージャは、すべてのチェックが完了し、接続が有効であることが確認された後で呼び出されます。login_procedure に指定したプロシージャはイベント接続では実行されません。

新規プロシージャを作成し、その新規プロシージャを呼び出すための login_procedure を設定して、デフォルト・データベース・オプション設定をカスタマイズできます。このカスタム・プロシージャは、sp_login_environment を呼び出すか、TDS 接続が確立されたことを検出し (デフォルトの sp_login_environment コードを確認します)、sp_tsql_environment を直接呼び出す必要があり

ます。この操作が失敗した場合、TDS ベースの接続は切断されることがあります。
sp_login_environment または sp_tsql_environment は編集しないでください。

参照

- 「post_login_procedure [データベース]」 481 ページ
- 「sp_login_environment システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
- 「sp_tsql_environment システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
- 「パスワードのセキュリティの強化」 923 ページ

例

次に、INVALID_LOGON エラーを通知して接続を拒否するサンプル・コードを示します。

```
CREATE PROCEDURE DBA.login_check()
  BEGIN
    DECLARE INVALID_LOGON EXCEPTION FOR SQLSTATE '28000';
    // Allow a maximum of 3 concurrent connections
    IF( DB_PROPERTY('ConnCount') > 3 ) THEN
        SIGNAL INVALID_LOGON;
    ELSE
        CALL sp_login_environment;
    END IF;
  END
  go


  GRANT EXECUTE ON DBA.login_check TO PUBLIC
  go
  SET OPTION PUBLIC.login_procedure='DBA.login_check'
  go
```

接続を禁止する代替方法の詳細については、「RAISERROR 文 [T-SQL]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

次の例は、ユーザの失敗した接続の数が 30 分間で 3 回よりも多くなった場合に、接続試行をブロックする方法を示しています。ブロック期間中にブロックされた試行は、すべて無効パスワード・エラーを受け取り、ログに失敗として記録されます。DBA がログを解析するために、ログは十分な時間保持されます。

```
CREATE TABLE DBA.ConnectionFailure(
   pk INT PRIMARY KEY DEFAULT AUTOINCREMENT,
   user_name CHAR(128) NOT NULL,
   tm TIMESTAMP NOT NULL DEFAULT CURRENT TIMESTAMP
)
go

CREATE INDEX ConnFailTime ON DBA.ConnectionFailure(
   user_name, tm )
go

CREATE EVENT ConnFail TYPE ConnectFailed
HANDLER
BEGIN
   DECLARE usr CHAR(128);
   SET usr = event_parameter( 'User' );

   // Put a limit on the number of failures logged.
   IF (SELECT COUNT(*) FROM DBA.ConnectionFailure
```



```
        WHERE user_name = usr
        AND tm >= DATEADD( minute, -30,
           CURRENT TIMESTAMP )) < 20 THEN
        INSERT INTO DBA.ConnectionFailure( user_name )
           VALUES( usr );

        COMMIT;
        // Delete failures older than 7 days.
        DELETE DBA.ConnectionFailure
        WHERE user_name = usr
        AND tm < dateadd( day, -7, CURRENT TIMESTAMP );
        COMMIT;
    END IF;
END
go

CREATE PROCEDURE DBA.login_check()
BEGIN
    DECLARE usr CHAR(128);
    DECLARE INVALID_LOGON EXCEPTION FOR SQLSTATE '28000';
    SET usr = CONNECTION_PROPERTY( 'Userid' );
    // Block connection attempts from this user
    // if 3 or more failed connection attempts have occurred
    // within the past 30 minutes.

    IF (SELECT COUNT(*) FROM DBA.ConnectionFailure
        WHERE user_name = usr
        AND tm >= DATEADD( minute, -30,
           CURRENT TIMESTAMP ) ) >= 3 THEN
        SIGNAL INVALID_LOGON;
    ELSE
        CALL sp_login_environment;
    END IF;
END
go


GRANT EXECUTE ON DBA.login_check TO PUBLIC
go

SET OPTION PUBLIC.login_procedure='DBA.login_check'
go
```

materialized_view_optimization オプション [データベース]

クエリへの応答を効率化するためにオプティマイザが実体化ビュー (Materialized View) を使用する方法を指定します。

指定可能な値

Disabled、**Fresh**、**Stale**、*N* { **Minute**[**s**] | **Hour**[**s**] | **Day**[**s**] | **Week**[**s**] | **Month**[**s**] }

デフォルト

Stale

スコープ

個々の接続、個々のユーザ、または PUBLIC グループに対して設定できます。すぐに有効になります。

備考

materialized_view_optimization オプションでは、オプティマイザが古い実体化ビュー (Materialized View) を使用できる状況を指定できます。

実体化ビュー (Materialized View) 内のデータは、そのビューが参照するベース・テーブルのデータが更新されることによって古くなります。実体化ビュー (Materialized View) のリフレッシュ頻度を設定するときには、データの古さをどの程度まで許容できるかを検討する必要があります。また、リフレッシュ・プロセス中はビューがクエリに応答できないため、ビューのリフレッシュに要する時間も考慮に入れます。さらに、データベースの現在の状態を反映していない結果が返されることを受け入れられるかどうかについても考慮する必要があります。このオプションは、以下のいずれかに設定できます。

- **Disabled** クエリの最適化に実体化ビュー (Materialized View) を使用しません。
- **Fresh** 古くなっていない (基本となるテーブルが前回のリフレッシュ以降に変更されていない) 場合のみ実体化ビュー (Materialized View) を使用します。
- **Stale** 古くなっていても実体化ビュー (Materialized View) を使用します。これがデフォルト設定です。
- **N { Minute[s] | Hour[s] | Day[s] | Week[s] | Month[s] }** 古い実体化ビュー (Materialized View) が指定された時間内にリフレッシュされている場合にかぎり、新しい実体化ビュー (Materialized View) と古い実体化ビュー (Materialized View) を使用します。分で指定する場合、2^{31} 分未満であることが必要です。データベース・サーバは、1 週間を 7 日、1 か月を 30 日と見なします。

クエリが実体化ビュー (Materialized View) を直接参照する場合は、古くなっているかどうかにかかわらず、そのビューが使用されます。この場合、materialized_view_optimization オプションは効力を持ちません。

max_client_statements_cached オプション [データベース]

クライアントでキャッシュされる文の数を制御します。

指定可能な値

整数 (0 ー 100)

スコープ

個々の接続または PUBLIC グループに設定できます。値を変更すると、すぐに有効になります。

デフォルト

10

備考

クライアントで文をキャッシュすると、同一の SQL 文が複数回準備されたときに、データベース要求と文の準備が減少します。同じ SQL 文の準備と削除が繰り返し行われると、クライアントは、文がデータベース・サーバに残された状態で、その文をキャッシュします。この文がアプリケーションによって削除された後でも、継続してキャッシュします。文のキャッシュにより、

データベース・サーバは、文の削除や再準備などの余分な作業を節約できます。スキーマの変更、データベース・オプション設定の変更、DROP VARIABLE 文の実行が生じると、準備文は自動的に削除され、その SQL 文が次に実行されるときに再び準備されます。これにより、不正な動作の原因となり得るような、キャッシュされた文が再使用されることのないようにします。

このオプションは、準備 (キャッシュ) されたまま維持できる文の最大数を指定します。キャッシュされた文は、max_statement_count リソース・ガバナーでは考慮されません。

このオプションの設定は、Embedded SQL、ODBC、OLE DB、ADO.NET、iAnywhere JDBC ドライバを使用して作成された接続に適用されます。Open Client、jConnect、HTTP 接続には適用されません。

このオプションに 0 を設定すると、クライアントの文のキャッシュが無効になります。この値を増加すると、アプリケーションが同じ SQL 文について 10 回以上にわたって準備と削除を繰り返す場合に、パフォーマンスが向上する可能性があります。たとえば、25 の SQL 文をスルーするループを処理するアプリケーションについて考えてみます。ループをスルーするごとに準備と削除を繰り返し、これらの SQL 文のそれぞれにまったく同じ文が含まれているときは、このオプションを 25 に設定すると、パフォーマンスが向上します。

このオプションの値を増加させると、クライアントにおけるメモリの使用量が増加し、サーバに対するキャッシュ要求が高まります。キャッシュされた文が大量になった場合で、スキーマの変更やオプション設定の変更によって文を再使用できなくなると、その接続において、文のキャッシュは自動的に無効になります。文のキャッシュが自動的に無効にされると、クライアントは再び定期的に文のキャッシュを実行して処理方法を再評価し、文のキャッシュを再び有効にすることに効果があるかどうかを判断します。

参照

- 「max_statement_count オプション [データベース]」 467 ページ
- ClientStmtCacheHits and ClientStmtCacheMisses プロパティ：「接続レベルのプロパティ」 518 ページ
- ClientStmtCacheHits and ClientStmtCacheMisses プロパティ：「サーバ・レベルのプロパティ」 540 ページ

max_cursor_count オプション [データベース]

接続で一度に使用できるカーソルの最大数を制限するリソース・ガバナーを制御します。

指定可能な値

整数

デフォルト

50

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。*どの*接続に対しても、このオプションを設定するには DBA パーミッションが必要です。

備考

このリソース・ガバナーを使用すると、接続ごとにユーザが使用できるカーソルの数を、DBA が制限できます。接続の制限を超える操作が行われると、リソースのガバナーを超過していることを示すエラーを出力します。

接続がストアド・プロシージャを実行する場合、プロシージャはプロシージャ所有者のパーミッションのもとで実行されます。ただし、プロシージャが使用するリソースは、現在の接続に割り当てられています。

オプションを 0 (ゼロ) に設定すると、リソース制限を削除できます。

max_plans_cached オプション [データベース]

キャッシュに格納される実行プランの最大数を指定します。

指定可能な値

整数

デフォルト

20

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。PUBLIC グループに対して、このオプションを設定するには DBA パーミッションが必要です。

備考

このオプションは、各接続でキャッシュされるプランの最大数を指定します。オプティマイザは、ストアド・プロシージャ、関数、トリガの中で実行されるクエリ、INSERT、UPDATE、DELETE の各文の実行プランをキャッシュします。ある接続でストアド・プロシージャ、ストアド関数、またはトリガに含まれる文が複数回実行された後、オプティマイザは、その文の再使用可能なプランを構築します。

再利用可能なプランでは、選択性推定やリライト最適化にホスト変数の値は使用されません。この結果、文の最適化が再度行われた場合に比べ、再使用可能なプランのコストの方が高くつく可能性があります。再使用可能なプランのコストが文に最適と思われるコストに近いとき、オプティマイザはそのプランをプラン・キャッシュに追加します。

このキャッシュは、CREATE TABLE や DROP TABLE など、テーブル・スキーマを変更する文が実行されたときにクリアされます。宣言されたテンポラリ・テーブルを参照する文はキャッシュされません。

このオプションに 0 を設定すると、プランのキャッシュが無効になります。

参照

- 「プランのキャッシュ」『SQL Anywhere サーバ - SQL の使用法』
- 「接続レベルのプロパティ」 518 ページ

max_query_tasks オプション [データベース]

データベース・サーバがクエリの並列処理に使用できるサーバ・タスクの最大数を指定します。

指定可能な値

整数

デフォルト

0

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

max_query_tasks option は、SQL 文で使用できる並列処理の最大レベルを設定します。このオプションは、クエリの並列処理に使用できるデータベース・サーバ・タスクの数を設定します。デフォルト値は 0 です。この場合、データベース・サーバは必要と判断した数のタスクを使用します。max_query_tasks オプションに 0 以外の値を指定した場合は、1 つのクエリについて使用可能なタスクの最大数が設定されます。max_query_tasks オプションを 1 に設定すると、クエリ内並列処理が無効になります。

サーバ・タスク、スレッド、クエリ実行の詳細については、「SQL Anywhere でのスレッド」 23 ページと「データベース・サーバのマルチプログラミング・レベルの設定」 27 ページを参照してください。

すべての要求に対してデータベース・サーバが使用できるタスクの数は、起動時に -gn オプションで設定されたしきい値によって制限されます。この数は、サーバがサービスを提供しているすべてのデータベースと接続に対してグローバルな最大数です。1 つの要求に使用されるタスクの数も、データベース・サーバで利用できる論理プロセッサの数によって制限されます。たとえば、-gtc オプションでプロセッサの同時実行性を 1 に設定した場合、クエリ内並列処理は無効となります。

クエリ内並列処理が有効である場合は、特定の条件を満たす SELECT 文がクエリ内並列処理によって処理されます。このクエリのアクセス・プランでは、クエリ内並列処理が使用されたことが交換演算子によって示されます。

各文に OPTION 句を含めることによって、テンポラリ設定とパブリック設定よりも、それぞれの INSERT、UPDATE、DELETE、または SELECT 文におけるこのオプションの設定を優先することができます。次の項を参照してください。

- 「INSERT 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「UPDATE 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「DELETE 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「SELECT 文」 『SQL Anywhere サーバ - SQL リファレンス』

参照

- 「-gn サーバ・オプション」 172 ページ
- 「-gt サーバ・オプション」 174 ページ

- 「-gtc サーバ・オプション」 175 ページ
- 「クエリ実行時の並列処理」 『SQL Anywhere サーバ - SQL の使用法』
- 「サーバ・レベルのプロパティ」 540 ページ

max_recursive_iterations オプション [データベース]

再帰共通テーブル式が反復できる最大回数を制限します。

指定可能な値

整数

デフォルト

100

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。PUBLIC グループに対して、このオプションを設定するには DBA パーミッションが必要です。

備考

計算が失敗して終了することが指定回数内で繰り返された場合に、再帰共通テーブル式の計算がアボートしてエラーが生成されます。再帰サブクエリは、反復が発生するたびに必要なリソースの総量が幾何学的に増加することがあります。このオプションを設定することによって、無限再帰が検出されるまでに消費される時間とリソースの量を制限し、しかも再帰共通テーブル式を意図した通りに動作させることができます。

このオプションに 0 を設定すると、再帰共通テーブル式が無効になります。

参照

- 「共通テーブル式」 『SQL Anywhere サーバ - SQL の使用法』

max_statement_count オプション [データベース]

接続で一度に使用できる準備文の最大数を制限するリソース・ガバナーを制御します。

指定可能な値

整数

デフォルト

50

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。*どの*接続に対しても、このオプションを設定するには DBA パーミッションが必要です。

備考

準備文を使用するアプリケーションは、準備文が不要になった後で明示的に削除されなかった場合、「準備文のリソース・ガバナーが制限を超えています。」というエラーを受け取ることがあります。max_statement_count オプションはリソース・ガバナーであり、これを使用することにより、DBA は 1 つの接続で使用される準備文の数を制限できます。接続の制限を超える操作が行われると、リソースのガバナーを超過していることを示すエラーを出力します。

接続がストアド・プロシージャを実行する場合、プロシージャはプロシージャ所有者のパーミッションのもとで実行されます。ただし、プロシージャが使用するリソースは、現在の接続に割り当てられています。

データベース・サーバには、接続によって作成される準備文ごとにデータ構造体が用意されます。アプリケーションがサーバに準備文が不要になったことを通知するか、接続が切断されるまで、これらの構造体は解放されません。接続ごとの準備文の数を減らすためには、DROP STATEMENT 要求に相当する処理を実行する必要があります。次の表は、SQL Anywhere でサポートされる API で実行できるコマンドを示しています。

インタフェース	文
ADO	RecordSet.Close
ADO.NET	SADataReader.Close または SADataReader.Dispose
Embedded SQL	DROP STATEMENT SQLFreeStmt(hstmt, SQL_DROP) または SQLFreeHandle(SQL_HANDLE_STMT, hstmt) resultSet.Close または Statement.Close RecordSet.Close SADataReader.Close または SaDataReader.Dispose
Java	resultSet.Close、Statement.Close
ODBC	SQLFreeStmt(hstmt, SQL_DROP) または SQLFreeHandle (SQL_HANDLE_STMT, hstmt)

注意
Java と .NET では、文を明示的に削除することをおすすめします。言語ルーチンは文リソースの割り付けを解除するサーバ呼び出しを発行しないため、ガーベジ・コレクションを使用して文を削除しないでください。また、ガーベジ・コレクション・ルーチンが実行される時点について保証はありません。

サーバにおいて接続のいずれかの時点でデフォルトの最大数を超える準備文のサポートが必要になった場合は、max_statement_count をデフォルトより大きな値に設定する必要があります。ただし、アクティブな準備文が多くなると、サーバ・メモリの使用量が増加します。
max_statement_count オプションを 0 (ゼロ) に設定し、準備文リソース・ガバナーを無効にすることもできますが、この方法は推奨できません。このようにすると、アプリケーションが準備文を適切に解放しない場合、サーバがメモリ不足によって異常終了しやすくなります。

参照

- 「文の準備」 『SQL Anywhere サーバ - プログラミング』
- 「DROP STATEMENT 文 [ESQL]」 『SQL Anywhere サーバ - SQL リファレンス』

max_temp_space オプション [データベース]

1 つの接続で使用できるテンポラリ・ファイル領域の最大サイズを設定します。

指定可能な値

Integer [**k** | **m** | **g** | **p**]

デフォルト

0

スコープ

すぐに有効になります。このオプションを設定するには、DBA パーミッションが必要です。

備考

このオプションでは、1 つの接続で使用できるテンポラリ・ファイル領域の最大サイズを指定できます。このテンポラリ・ファイル領域の最大サイズを超えると、要求は失敗します。max_temp_space オプションが効力を持つためには、temp_space_limit_check オプションを On (デフォルト) に設定する必要があります。

デフォルト値の 0 は、1 つの接続が要求できるテンポラリ・ファイル領域のサイズに制限がないことを意味します。0 以外の値は、1 つの接続で使用できるテンポラリ・ファイル領域のバイト数を示します。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ **k**、**m**、**g** を使用します。**p** を使用すると、値はテンポラリ・ファイル領域の総量に対する割合を表します。

テンポラリ・ファイル領域を要求する接続では、データベース・サーバが max_temp_space の設定をチェックし、要求がテンポラリ・ファイル領域の最大サイズを超えていないことを確認します。接続が最大サイズを超えるテンポラリ・ファイル領域を要求すると、その要求は失敗し、エラー SQLSTATE_TEMP_SPACE_LIMIT が生成されます。

参照

- 「temp_space_limit_check オプション [データベース]」 503 ページ
- 「sa_disk_free_space システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』

例

次の文は接続に対して 1 GB の制限を設定します。

```
SET OPTION PUBLIC.max_temp_space = '1g';
```

次の 2 つの文は、いずれも接続に対して 1 MB の制限を設定します。

```
SET OPTION PUBLIC.max_temp_space = 1048576;
SET OPTION PUBLIC.max_temp_space = '1m';
```

次の文は、テンポラリ・ファイル領域の使用を全体の 5 % に制限します。

```
SET OPTION PUBLIC.max_temp_space = '5p';
```

min_password_length オプション [データベース]

新しいパスワードの最小長をデータベースに設定します。

指定可能な値

整数

値はバイト数で指定します。シングルバイト文字セットの場合、これは文字数と同じになります。

デフォルト

0 文字

スコープ

PUBLIC グループのみに設定できます。すぐに有効になります。このオプションを設定するには、DBA パーミッションが必要です。

備考

このオプションを使用すると、データベース管理者は、セキュリティを強化するために、新しいパスワードすべてに最小長を設定できます。既存のパスワードは影響を受けません。パスワードは最大長が 255 バイトで、大文字と小文字が区別されます。

参照

- 「verify_password_function オプション [データベース]」 513 ページ

例

新しいパスワードの最小長を 6 バイトに設定します。

```
SET OPTION PUBLIC.min_password_length = 6;
```

nearest_century オプション [互換性]

文字列から日付への変換で、2 桁の年の解釈を制御します。

指定可能な値

整数 (0 ー 100)

デフォルト

50

備考

このオプションは、文字列から日付またはタイムスタンプに変換するときに、2 桁の年の処理を制御します。

nearest_century 設定は、ロールオーバ・ポイントとして動作する数値です。この値より小さい 2 桁の年は 20yy に変換され、この値以上の年は 19yy に変換されます。

従来の SQL Anywhere では、年に 1900 を加算していました。Adaptive Server Enterprise では最も近い世紀を使用するので、*yy* が 50 より小さい場合は 20yy になります。

non_keywords オプション [互換性]

個々のキーワードをオフにして、識別子として使用できるようにします。

指定可能な値

文字列

デフォルト

空の文字列

備考

このオプションは、特定の製品リリース以降に作成された個々のキーワードまたはすべてのキーワードをオフにします。これにより、古いバージョンの製品で作成されたアプリケーションが新しいキーワードで破損されないことが保証されます。現在データベースにキーワードである識別子がある場合、すべてのアプリケーションまたはスクリプトで識別子を二重引用符で囲むか、non_keywords オプションを使用してキーワードをオフにできます。

個々のキーワードの指定に加え、次に示すキーワード・リストの中から 1 つの特別値を使用して、特定のリリース以降のすべてのキーワードをオフにできます。

```
keywords_4_0_d, keywords_4_0_c, keywords_4_0_b, keywords_4_0_a, keywords_4_0,
keywords_5_0_01, keywords_5_0
```

次の文を記述すると、TRUNCATE と SYNCHRONIZE がキーワードとして認識されません。

```
SET OPTION non_keywords = 'TRUNCATE, SYNCHRONIZE';
```

次の文を記述すると、バージョン 4.0.d 以降に作成されたすべてのキーワードがキーワードとして認識されません。

```
SET OPTION non_keywords = 'keywords_4_0_d';
```

このオプションで新規設定を行うと、前に設定しているものに置き換わります。次の文は以前の設定内容をすべてクリアします。

```
SET OPTION non_keywords =;
```

このオプションを使用すると、オフにしたキーワードを使用する SQL 文を使えなくなります。そのような SQL 文を指定すると、構文エラーが生じます。

参照

- 「キーワード」『SQL Anywhere サーバ - SQL リファレンス』

odbc_describe_binary_as_varbinary [データベース]

SQL Anywhere ODBC ドライバの BINARY カラムの記述方法を制御します。

指定可能な値

On、Off

デフォルト

Off

備考

このオプションを使用すると、すべての BINARY と VARBINARY カラムをアプリケーションに対して BINARY か VARBIARY のどちらで記述するかを選択できます。デフォルトでは、SQL Anywhere ODBC ドライバは BINARY と VARBINARY の両方のカラムを SQL_BINARY として記述します。このオプションを On に設定すると、ODBC ドライバは BINARY と VARBINARY カラムを SQL_VARBINARY として記述します。このオプションの設定にかかわらず、BINARY カラムと VARBINARY カラムを区別することはできません。

BINARY カラムは常に 0 埋め込みで、VARBINARY カラムはそうではない Delphi アプリケーションを使用する場合は、このオプションを On に設定することをおすすめします。このオプションを On にしてすべてのカラムが可変長データ型として扱われるようにすると、Delphi のパフォーマンスが向上します。

参照

- 「BINARY データ型」『SQL Anywhere サーバ - SQL リファレンス』
- 「VARBINARY データ型」『SQL Anywhere サーバ - SQL リファレンス』

odbc_distinguish_char_and_varchar オプション [データベース]

SQL Anywhere ODBC ドライバの CHAR カラムの記述方法を制御します。

指定可能な値

On、Off

デフォルト

Off

備考

接続を開くと、SQL Anywhere ODBC ドライバは、このオプションの設定を使用して CHAR カラムの記述方法を決定します。このオプションを Off (デフォルト) に設定すると、CHAR カラムは SQL_VARCHAR として記述されます。このオプションを On に設定すると、CHAR カラムは SQL_CHAR として記述されます。VARCHAR カラムは、常に SQL_VARCHAR として記述されます。

odbc_distinguish_char_and_varchar オプションは、NCHAR カラムが SQL_WCHAR または SQL_WVARCHAR として記述されるかどうかも制御できます。このオプションを Off に設定すると、NCHAR カラムは SQL_WVARCHAR として記述されます。このオプションを On に設定すると、NCHAR カラムは SQL_WCHAR として記述されます。NVARCHAR カラムは、常に SQL_WVARCHAR として記述されます。

参照

- 「NCHAR データ型」『SQL Anywhere サーバ - SQL リファレンス』
- 「NVARCHAR データ型」『SQL Anywhere サーバ - SQL リファレンス』

oem_string オプション [データベース]

データベース・ファイルのヘッダ・ページ内にあるユーザ指定の情報を保存します。

指定可能な値

文字列 (最大 128 バイト)

デフォルト

空の文字列

スコープ

PUBLIC グループのみに設定できます。すぐに有効になります。このオプションを設定するには、DBA パーミッションが必要です。

備考

データベース・ファイルのヘッダ・ページ内の情報を保存しておき、アプリケーションから直接ファイルを読み込んで情報を抽出できます。このページはシステム DB 領域ファイルのヘッダに保存されます。OEM 文字列に対して 128 バイトを超える値を指定すると、エラーが返されます。

スキーマ・バージョン、アプリケーション名、アプリケーションのバージョンなどの情報は、保存しておくと役に立つことがあります。また、アプリケーションがデータベース・ファイルを使用する前に検証のために読み込む文字列を保存することによって、アプリケーションはデータベースを起動しなくても OEM 文字列によってデータベース・ファイルがそのアプリケーションに関連付けられているかどうかを確認できます。つまり、アプリケーションを設計するときに、OEM 文字列を使用してデータベース・ファイルがそのアプリケーション用であることを検証するプロセスを組み込むことができます。さらに、ユーザに表示するメタデータを抽出することもできます。

oem_string をシステム DB 領域ファイルのヘッダ内に設定するには、次の文を実行します。

```
SET OPTION PUBLIC.oem_string=user-specified-string;
```

user-specified-string の値は、ISYSOPTIONS システム・テーブルとシステム DB 領域ファイルのヘッダの両方に保存されます。この文字列が SET OPTION 文に渡されるときには変換が行われないため、文字列を必要な文字セットで定義してから SET OPTION 文に指定する必要があります。文字列を必要な文字セットに変換するには、CSCONVERT 関数を使用します。

oem_string の値を問い合わせるには、以下の方法があります。

- oem_string 接続プロパティを使用する。

  ```
  SELECT CONNECTION_PROPERTY( 'oem_string' );
  ```

- SYSOPTION システム・ビューを使用する。

```
SELECT setting FROM SYSOPTION WHERE "option" = 'oem_string';
```

♦ **アプリケーションから oem_string オプションの値を問い合わせるには**

1. データベース・システム DB 領域ファイルを開きます。
2. このファイルの最初のページをバッファに読み込みます。
3. バッファから OEM 文字列の前後にある 2 バイトのプレフィクスとサフィックスを検索します。

   プレフィクスとサフィックスは、*sqldef.h* にそれぞれ DB_OEM_STRING_PREFIX と DB_OEM_STRING_SUFFIX として定義されています。これら 2 つの文字列に囲まれたすべてのバイトがデータベースに定義された OEM 文字列です。

SQL Anywhere には 2 つのサンプル・プログラムが付属しており、*oem_string* ディレクトリに保存されています。

- *dboem.cpp* は、OEM 文字列を抽出しサーバ・メッセージ・ウィンドウに出力する方法を示す C プログラムです。
- *dboem.pl* は、OEM 文字列を抽出し PERL スクリプト内 の stdout に出力する方法を示します。

警告
アプリケーションからデータベース内の OEM 文字列に直接書き込みを行うことはできません。そのようにすると、データベースのヘッダ・ページが破損します。

Windows では、サーバがこのデータベース・ファイルをロードした場合、アプリケーションからこのファイルを直接読み込むことはできません。データベース・サーバは、このファイルに排他ロックを設定します。これに対し、サポートされる UNIX プラットフォームでは、読み込みパーミッションのあるアプリケーションであれば、このファイルをいつでも直接読み込むことができます。ただし、OEM 文字列に対する変更は、直ちにファイルに反映されないことがあります。チェックポイントを発行すると、データベース・サーバがページ 0 をディスクにフラッシュするため、現在の OEM 文字列値がファイルに反映されます。

OEM 文字列を変更してから次のチェックポイントまでの間にデータベース・サーバで障害が発生すると、ファイルのヘッダに新しい OEM 文字列値が反映されないことがあります。新しい OEM 文字列値は、データベースのリカバリが完了した後で正しく設定されます。

参照

- 「CSCONVERT 関数 [文字列]」 『SQL Anywhere サーバ - SQL リファレンス』

例

次の例は、データベース・ファイルに関する情報を示す OEM 文字列を暗号化し、データベース・ヘッダ・ファイルに保存します。

```
SET OPTION PUBLIC.oem_string=
 BASE64_ENCODE(
   ENCRYPT( 'Database version 10', '4j67lm3z' ) );
```

OEM 文字列の値を取得するには、次の文を使用します。

```
SELECT DECRYPT(
  BASE64_DECODE(
  CONNECTION_PROPERTY( 'oem_string' ), '4j67lm3z' ) );
```

on_charset_conversion_failure オプション [データベース]

文字の変換中にエラーが発生した場合の動作を制御します。

指定可能な値

Ignore、Warning、Error

デフォルト

Ignore

備考

文字の変換中にエラーが発生した場合の動作を次のように制御します。

- **Ignore**　エラーも警告も表示しません。
- **Warning**　置換と不正な文字列を警告としてレポートします。不正な文字列は変換されません。
- **Error**　置換と不正な文字列をエラーとしてレポートします。

クライアントとデータベース間で文字セットの変換が必要な場合、不正な文字が検出されたり文字の置換が使用されたりしたときに、それを無視するか、警告を返すか、エラーを返すかをこのオプションで制御します。

シングルバイトからシングルバイトへの変換では、置換と不正な文字のレポートはできないので、Ignore に設定する必要があります。

このオプションは、損失を伴う変換がクライアントで発止した場合の動作は制御しません。たとえば、クライアントからの SQL 文は CHAR データベース文字セットに格納されているか、変換される必要があります。ユニコードのクライアント・アプリケーションが SQL 文を準備し、その文に CHAR データベース文字セットで表現できない文字が含まれているとします。この場合、置換文字が代わりに使用されます。ただし、損失を伴う変換がクライアントで発生したため、データベース・サーバではこの変換が発生したことがわかりません。

参照

- 「CHAR と NCHAR の比較」『SQL Anywhere サーバ - SQL リファレンス』
- 「NCHAR から CHAR への変換」『SQL Anywhere サーバ - SQL リファレンス』
- 「置換文字」『SQL Anywhere サーバ - SQL リファレンス』

on_tsql_error オプション [互換性]

ストアド・プロシージャのエラー処理を制御します。

指定可能な値

文字列 (指定可能な値は以下を参照)

デフォルト

Conditional

jConnect 接続の場合は Continue

備考

このオプションは、ストアド・プロシージャのエラー処理を制御します。

- **Stop**　エラーの検出と同時に実行を停止します。
- **Conditional**　プロシージャが ON EXCEPTION RESUME を使用していて、エラーのすぐ後ろの文がエラーを処理する場合は継続します。それ以外の場合は終了します。
- **Continue**　次に続く文に関係なく実行は継続されます。複数のエラーがある場合は、ストアド・プロシージャで最初に検出されたエラーが返されます。

on_tsql_error の設定 Conditional と Continue は、いずれも Adaptive Server Enterprise との互換性を確保するために使用します。Continue が Adaptive Server Enterprise 動作を最も忠実にシミュレートします。特に、新しい Transact-SQL ストアド・プロシージャを開発している場合は、エラーがより迅速にレポートされるため、Conditional に設定することをおすすめします。

このオプションを Stop または Continue に設定した場合は、その設定が continue_after_raiserror オプションの設定より優先されます。ただし、このオプションを Conditional (デフォルト) に設定した場合は、RAISERROR 文の後の動作は continue_after_raiserror オプションによって決まります。

参照

- 「CREATE PROCEDURE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「CREATE PROCEDURE 文 [T-SQL]」『SQL Anywhere サーバ - SQL リファレンス』
- 「Transact-SQL のプロシージャ言語の概要」『SQL Anywhere サーバ - SQL の使用法』
- 「continue_after_raiserror オプション [互換性]」 437 ページ

optimistic_wait_for_commit オプション [互換性]

wait_for_commit オプションのロック動作を制御します。

指定可能な値

On、Off

デフォルト

Off

備考

デフォルトでは、optimistic_wait_for_commit は Off です。

ロック動作は、optimistic_wait_for_commit と wait_for_commit が On の場合、次のように変更されます。

- 一致するプライマリ・キー・ローなしで外部キー・ローを追加する場合、プライマリ・キー・ローでロックが実行されません。
- トランザクションがプライマリ・ローを追加した場合にこれがコミット可能になるのは、プライマリ・ローが追加されるときにトランザクションでプライマリ・ローを参照するすべての外部ローに排他ロックがある場合のみです。

この機能は、バージョン 5.x.x アプリケーションをバージョン 8.0.x 以降へ移行するのに使用します。これは、(2 つのトランザクションが同時に複数の外部ローを同一キーで追加しないかぎり) トランザクションが外部ローを追加してからプライマリ・ローを追加するときに、バージョン 5.x.x のロック動作を模倣することで可能になります。

トランザクションがコミットされない状況が数多くあるため、このオプションを日常的に使用することはおすすめしません。そのような状況には次のようなものがあります。

- プライマリ・ローが存在しないときに複数のトランザクションが同一キーを使用して同時に複数の外部ローを追加する場合、1 トランザクション (対応するプライマリ・ローを追加したトランザクション) のみがコミットされます。
- 1 つのトランザクションがプライマリ・ローを削除してそれを追加し直す場合、そのほとんどがコミットされません (ただし、一致するすべての外部ローの排他ロックを取得できればコミットされます)。

参照

- 「wait_for_commit オプション [データベース]」 515 ページ

optimization_goal オプション [データベース]

クエリ処理の最適化の対象を、最初のローを迅速に返すこと、または完全な結果セットを返すコストを最小限に抑えることのどちらかに指定します。

指定可能な値

First-row または All-rows

デフォルト

All-rows

備考

optimization_goal オプションは、SQL Anywhere において SQL データ操作言語 (DML) 文を応答時間に対して最適化するか、リソースの総消費量に対して最適化するかを制御します。

このオプションを All-rows (デフォルト) に設定すると、SQL Anywhere はクエリを最適化して、予測される最短の合計検索時間でアクセス・プランを選択します。PowerBuilder DataWindow アプリケーションなど、処理の前に結果セット全体が必要になるアプリケーションでは、optimization_goal を All-rows に設定するのが適切です。All-rows の設定では、カーソルが開いた

ときに結果全体が実体化されるため、insensitive (ODBC の静的) カーソルにも適しています。また、結果セットのスクロールを目的とするスクロール (ODBC キーセット駆動型) カーソルにも適しています。

このオプションを First-row に設定すると、SQL Anywhere は、クエリの結果の最初のローをフェッチするまでの時間を短縮するアクセス・プランを選択します。この場合、検索にかかる合計時間は長くなることがあります。また、通常 SQL Anywhere オプティマイザでは、可能であれば結果の実体化を必要とするアクセス・プランは使用しないで、最初のローを返すまでの時間を短縮します。この設定では、オプティマイザは、明示的なソートの操作を必要とするアクセス・プランではなく、クエリの ORDER BY 句を満たすインデックスを使用するアクセス・プランを採用します。

クエリの FROM 句の FASTFIRSTROW テーブル・ヒントを使用すると、特定のクエリの最適化ゴールを First-row に設定できます。この場合、optimization_goal の設定を変更する必要はありません。

FASTFIRSTROW テーブル・ヒントの使用の詳細については、「FROM 句」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

各文に OPTION 句を含めることによって、テンポラリ設定とパブリック設定よりも、それぞれの INSERT、UPDATE、DELETE、または SELECT 文におけるこのオプションの設定を優先することができます。次の項を参照してください。

- 「INSERT 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「UPDATE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「DELETE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「SELECT 文」『SQL Anywhere サーバ - SQL リファレンス』

optimization_level オプション [データベース]

SQL Anywhere クエリ・オプティマイザが SQL 文のアクセス・プランの検索に費やす作業量を制御します。

指定可能な値

0-15

デフォルト

9

備考

optimization_level オプションは、SQL Anywhere オプティマイザが SQL データ操作言語 (DML) の最適化に費やす作業量を制御します。このオプションは、任意の SELECT ブロックについてオプティマイザが考慮する代替のジョイン方式の最大数を制御します。optimization_level の設定値が高いほど、オプティマイザが考慮するジョイン方式の最大数は大きくなります。

このオプションを 0 に設定すると、SQL Anywhere オプティマイザは実行のために考慮する最初のアクセス・プランを選択し、事実上、代替プランのコストベースの比較を避けることになります。さらに、レベル 0 では、ネストされたクエリのセマンティックな最適化が一部無効になりま

す。このオプションが 0 よりも大きい値に設定されると、オプティマイザは代替方式を評価し、予想コストが最も低いものを選択します。このオプションがデフォルトの 9 よりも大きい値に設定されると、オプティマイザは代替方式をより積極的に検索し、その結果、最適化フェーズで経過する時間がはるかに長くなる可能性があります。

代表的なシナリオでは、アプリケーションが DML 文に対してより速い OPEN 時間を必要とし、文が複雑であってもクエリの実行時間は非常に短かいためにオプティマイザによって選択された特定のアクセス・プランはあまり重要ではないことがわかっている場合は、このオプションは一時的に低いレベル (0、1、2 など) に設定されます。optimization_level の PUBLIC 設定をデフォルトから変更することはおすすめできません。

optimization_level オプションの設定の結果は、optimization_goal と optimization_workload オプションの設定とは無関係です。

単純な DML 文 (特定の行を識別する WHERE 句に等号条件を含んだ単一ブロック、単一テーブルのクエリ) はヒューリスティックに最適化されるため、コストベースのオプティマイザをすべてバイパスします。単純な DML 文の最適化は、optimization_level オプションの設定からは影響を受けません。オプティマイザ・バイパス・メカニズムによって最適化された要求の数は、QueryBypassed 接続プロパティとして使用できます。

QueryBypassed 接続プロパティの詳細については、「接続レベルのプロパティ」 518 ページを参照してください。

各文に OPTION 句を含めることによって、テンポラリ設定とパブリック設定よりも、それぞれの INSERT、UPDATE、DELETE、または SELECT 文におけるこのオプションの設定を優先することができます。次の項を参照してください。

- 「INSERT 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「UPDATE 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「DELETE 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「SELECT 文」 『SQL Anywhere サーバ - SQL リファレンス』

optimization_workload オプション [データベース]

クエリ処理において、更新と読み込みを組み合わせた負荷に対して最適化するか、または大部分が読み込みベースの負荷に対して最適化するかを決定します。

指定可能な値

Mixed、OLAP

デフォルト

Mixed

スコープ

PUBLIC グループのみに設定できます。DBA 権限が必要です。

備考

optimization_workload オプションは、SQL Anywhere が、更新と読み取りが組み合わさった負荷、または主に読み取りベースの負荷に対して、どちらのクエリ処理向けに最適化されるか制御します。

このオプションが Mixed (デフォルト) に設定されている場合、SQL Anywhere は短い挿入、更新、削除と、実行時間の長い読み取り専用クエリを組み合わせた負荷に対して適切なクエリ最適化アルゴリズムを選択します。

このオプションが OLAP に設定されている場合、SQL Anywhere は、実行時間の長いクエリの大部分とバッチ更新を組み合わせた負荷に対して適切なアルゴリズムを選択します。特に、オプティマイザは、クラスタード・ハッシュ Group By クエリ実行アルゴリズムを使用するように選択する場合があります。

オプションが OLAP に設定されている場合、クラスタード・ハッシュ Group By アルゴリズムが有効になります。このオプションを Mixed (デフォルト) に設定すると、このアルゴリズムは無効になります。

各文に OPTION 句を含めることによって、テンポラリ設定とパブリック設定よりも、それぞれの INSERT、UPDATE、DELETE、または SELECT 文におけるこのオプションの設定を優先することができます。次の項を参照してください。

- 「INSERT 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「UPDATE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「DELETE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「SELECT 文」『SQL Anywhere サーバ - SQL リファレンス』

参照

- 「クラスタード・ハッシュ GROUP BY アルゴリズム」『SQL Anywhere サーバ - SQL の使用法』

percent_as_comment オプション [互換性]

パーセント記号の解釈を制御します。

指定可能な値

On、Off

デフォルト

On

備考

% をコメント・マーカとして使用しないことをおすすめします。

バージョン 6 より前では、SQL 文内ではパーセント記号 (%) にかぎり、コメント・デリミタとして扱っていました。バージョン 5 以降では、*//*、*/* */*、*--* (ダブル・ハイフン) など、その他のコメント・マーカも使用できます。ダブル・ハイフン・スタイルは SQL/2003 コメント・デリミタです。

Adaptive Server Enterprise は、**%** をモジュロ演算子として処理し、SQL Anywhere の mod 関数をサポートしません。以前は、両方の環境で動作し、モジュロ演算を行う文を記述することはできませんでした。

percent_as_comment オプションは、% の意味を制御します。下位互換性を保つために、デフォルト設定は On になっています。Adaptive Server Enterprise との互換性を持たせるには、Off に設定してください。

% スタイルのコメントで作成されたプロシージャ、トリガ、ビューは、カタログに保管されるときにダブル・ハイフン・コメントに変換されます。

pinned_cursor_percent_of_cache オプション [データベース]

カーソルを固定するために使用できるキャッシュの割合を指定します。

指定可能な値

整数 (0 ー 100)

デフォルト

10

スコープ

PUBLIC グループのみに設定できます。DBA 権限が必要です。

備考

サーバは、カーソルの実装に必要なデータ構造を、仮想メモリのページに格納します。これらのページは、フェッチ要求から次のフェッチ要求までの間メモリ内にロックされているため、次のフェッチ要求を受信したときにすぐに使用できます。

メモリが少ない環境で、これらのページによって必要以上にキャッシュが占有されないように、カーソルの固定に使用するキャッシュの割合は制限されています。この制限を調整するには、pinned_cursor_percent_of_cache オプションを使用します。

このオプションの値は、0 ー 100 の割合 (%) で指定します。デフォルト値は 10 です。0 に設定すると、次のフェッチ要求までの間カーソル・ページは固定されません。

post_login_procedure [データベース]

ユーザが接続したときにアプリケーションが表示する必要のあるメッセージが結果セットに含まれるプロシージャを指定します。

指定可能な値

文字列

デフォルト

空の文字列

スコープ

DBA 権限が必要です。

備考

post_login_procedure を空の文字列以外に設定すると、アプリケーションは、このオプションに指定されたプロシージャを接続プロセス中に呼び出し、ユーザに対して表示するメッセージを特定できます。このオプションには、*owner.function-name* という形式の値を指定する必要があります。これによって、ユーザが関数を無効にすることを防止できます。

このオプションを設定すると、Sybase Central 用の SQL Anywhere プラグインである Interactive SQL と dbisqlc がプロシージャを呼び出し、プロシージャが返すメッセージをダイアログ内に表示します。SQL Anywhere に付属していないアプリケーションについては、必要に応じて、このオプションに指定されたプロシージャを呼び出してメッセージを表示するように変更を加えてください。

接続時にアプリケーションがメッセージを表示することが必要となる状況としては、パスワード有効期限システムを実装している場合に、パスワードの期限切れが近づいていることをユーザに通知することがあります。このような機能を利用すると、ユーザが接続するたびに、パスワードが期限切れとなることを数日前に知らせることができます。

このオプションに指定するプロシージャは、1 つ以上のローと 2 つのカラムで構成される結果セットを返す必要があります。1 つ目のカラムは VARCHAR(255) 型で、メッセージのテキスト (メッセージがない場合は NULL) を返します。2 つ目のカラムは INT 型で、アクションのタイプを返します。アクションとして指定可能な値は次のとおりです。

- **0** メッセージを表示 (メッセージがある場合)
- **1** メッセージを表示し、ユーザにパスワードの変更を要求
- **2-99** 予約
- **100 以上** ユーザ定義

SQL Anywhere プラグインの dbisql と dbisqlc は、アクションの値にかかわらず、NULL 以外であれば、すべてのメッセージを表示します。アクションを 1 に設定すると、SQL Anywhere プラグイン と dbisql (dbisqlc ではない) はユーザにパスワードを変更するよう要求し、新しいパスワードをユーザ指定の値に設定します。

post_login_procedure の使用と、パスワード有効期限の運用などの詳細なパスワード規則の実装を示す例については、「パスワード検証関数の使用」を参照してください。

参照

- 「login_procedure オプション [データベース]」 460 ページ
- 「パスワードのセキュリティの強化」 923 ページ

例

次の例では、p_post_login_check というプロシージャを使用して、ユーザにパスワードの期限切れが近づいていることを知らせ、パスワードを変更するよう要求します。

```
CREATE PROCEDURE DBA.p_post_login_check()
RESULT( message_text VARCHAR(255), message_action INT )
BEGIN
  DECLARE message_text        CHAR(255);
  DECLARE message_action      INT;

  -- assume the password_about_to_expire variable was set by the login procedure
  IF password_about_to_expire = 1 THEN
    SET message_text = 'Your password is about to expire';
    SET message_action = 1;
  ELSE
    SET message_text = NULL;
    SET message_action = 0;
  END IF;
  -- return message (if any) through this result set
  SELECT message_text, message_action;
END;

GRANT EXECUTE ON DBA.p_post_login_check TO PUBLIC;

SET OPTION PUBLIC.post_login_procedure = 'DBA.p_post_login_check';
```

precision オプション [データベース]

10 進法計算での結果の最大桁数を指定します。

指定可能な値

整数 (1 ー 127)

デフォルト

30

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

精度は、小数点の左右の合計桁数です。scale オプションは、計算結果が最大 precision にトランケートされた場合の、小数点以下の最小桁数を指定します。

掛け算、割り算、足し算、引き算、集合関数はすべて結果が最大精度を超えることができます。

たとえば、DECIMAL(8,2) と DECIMAL(9,2) を掛けると、結果には DECIMAL(17,4) が必要です。precision が 15 の場合、15 桁のみが結果に残ります。scale が 4 の場合、結果は DECIMAL(15,4) です。scale が 2 の場合、結果は DECIMAL(15,2) です。どちらの場合も、オーバフローの可能性があります。

prefetch オプション [データベース]

クライアント・アプリケーションで使用できるようになる前に、ローがクライアント側にフェッチされるかどうかを制御します。

指定可能な値

Off、Conditional、Always

デフォルト

Conditional

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

このオプションは、クライアント・アプリケーションで使用できるようになる前に、ローがクライアント・サイドにフェッチされるかどうかを制御します。一度にいくつかのローをフェッチすると、クライアント・アプリケーションが一度に 1 つのローを要求した場合 (カーソルのローをループする場合など) でも、応答時間が短縮され、データベースへの要求数の減少によって全体的なスループットも向上します。

- Off はプリフェッチが行われないことを意味します。
- Conditional (デフォルト) の場合は、カーソル・タイプが SENSITIVE か、クエリにプロキシ・テーブルが含まれないかぎり、プリフェッチが発生します。
- Always は、SENSITIVE カーソル・タイプの場合も、プロキシ・テーブルが関連するカーソルの場合も、プリフェッチが行われることを意味します。.

Always は、一部のカーソルのセマンティックに影響するため、注意して使用してください。たとえば、SENSITIVE タイプのカーソルが ASENSITIVE タイプになる場合があります。また、プリフェッチと、アプリケーションのフェッチ要求の間に値が更新された場合は、古い値がフェッチされることがあります。さらに、プロキシ・テーブルが関連するカーソルでプリフェッチを使用した場合、クライアントがプリフェッチ・ローを再フェッチしようとすると、エラー 668「カーソルは FETCH NEXT 操作に制限されています」が発生する可能性があります。最初のフェッチの後で、初めてフェッチ・カラムが再バインドまたはバインドされた場合、クライアントは、ロールバックの後、または 0 の相対フェッチ時、あるいは場合によっては GET DATA が使用されるときに、プリフェッチ・ローを再フェッチしようとすることがあります。

prefetch オプションの設定は、Open Client 接続と jConnect 接続では無視されます。

DisableMultiRowFetch 接続パラメータが YES に設定されている場合、prefetch データベース オプションは無視され、プリフェッチは行われません。

このオプションでは、以前は On も有効な値でした。この値は現在は Conditional のエイリアスです。

参照

- 「ローのプリフェッチ」 『SQL Anywhere サーバ - プログラミング』
- 「DisableMultiRowFetch 接続パラメータ [DMRF]」 245 ページ

preserve_source_format オプション [データベース]

プロシージャ、トリガ、ビュー、イベント・ハンドラの元のソース定義をシステム・ファイルに保存するかどうかを制御します。保存した場合は、SYSTAB、SYSPROCEDURE、SYSTRIGGER、SYSEVENT 内のカラム source に保存されます。

指定可能な値

On、Off

デフォルト

On

スコープ

PUBLIC グループのみに設定できます。DBA 権限が必要です。

備考

preserve_source_format を On に設定すると、サーバは、プロシージャ、ビュー、トリガ、イベントの CREATE 文と ALTER 文によってフォーマットされたソースを保存し、それを適切なシステム・ビューの送信元カラムに配置します。

フォーマットされていないソース・テキストは、同じシステム・テーブルの proc_defn、trigger_defn、view_defn の各カラムに格納されます。ただし、これらの定義は、Sybase Central では簡単には読めません。フォーマットされた source カラムでは、スペース、コメント、大文字または小文字を任意に選んで定義を参照できます。

このオプションを OFF にすると、データベースにオブジェクト定義を保存するために使用される領域を減らすことができます。このオプションは、PUBLIC ユーザのみに設定できます。

prevent_article_pkey_update オプション [データベース] [Mobile Link クライアント]

パブリケーションに関連するテーブルのプライマリ・キー・カラムの更新を制御します。

指定可能な値

On、Off

デフォルト

On

備考

このオプションを On に設定すると、パブリケーションに含まれるテーブルのプライマリ・キー・カラムの更新が禁止されます。このオプションを使用することで、特にレプリケーションと同期の環境におけるデータの整合性が確保されます。

> **警告**
> 同期環境やレプリケーション環境では、このオプションを Off に設定しないでください。

qualify_owners オプション [SQL Remote]

SQL Remote によってレプリケートされる SQL 文で、修飾されたオブジェクト名を使用するかどうかを指定します。

指定可能な値

On、Off

デフォルト

On

備考

SQL Anywhere のインストール環境で修飾が必要でない場合は、オプションを Off にすると、メッセージが少し小さくなります。

参照

- 「SQL Remote オプション」 『SQL Remote』

query_plan_on_open オプション [互換性]

カーソルを開くときに、プランが戻るかどうかを制御します。

指定可能な値

On、Off

デフォルト

Off

備考

以前のバージョンのソフトウェアでは、カーソルの OPEN を行うたびにサーバはクエリ・プラン (最大 70 バイト) を示す文字列を SQLCA **sqlerrmc** フィールドに戻していました。EXPLAIN 文か PLAN 機能を使用すると詳細な説明が得られます。このため、OPEN でのクエリ・プランの計算と戻りが必要になるのは、古いアプリケーションとの互換性を保つ場合のみです。
query_plan_on_open オプションは、OPEN 時にプランが戻るかどうかを制御します。

quote_all_identifiers オプション [SQL Remote]

SQL Remote によってレプリケートされる SQL 文で、引用符で囲んだ識別子を使用するかどうかを指定します。

指定可能な値

On、Off

デフォルト

Off

備考

このオプションを Off に設定すると、dbremote が SQL Anywhere で引用符が必要な識別子を (通常行われているように) 引用符で囲みます。

このオプションが On の場合、すべての識別子が引用符で囲まれます。

参照

- 「SQL Remote オプション」 『SQL Remote』

quoted_identifier オプション [互換性]

二重引用符内の文字列の解釈を制御します。

指定可能な値

On、Off

デフォルト

On

Open Client 接続と jConnect 接続の場合は Off

備考

このオプションは、二重引用符内の文字列を、識別子 (On) とリテラル文字列 (Off) のどちらとして解釈するかを制御します。quoted_identifier オプションは、Transact-SQL との互換性を保つために実装されています。

「Transact-SQL との互換性を維持するためのオプション設定」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

read_past_deleted オプション [データベース]

独立性レベル 1 または 2 でコミットされていない削除におけるサーバ動作を制御します。

指定可能な値

On、Off

デフォルト

On

備考

read_past_deleted が On (デフォルト) の場合、独立性レベル 1 または 2 の逐次スキャンではコミットされていない削除ローを省略します。Off の場合、(削除トランザクションがコミットまたはロールバックするまで) 逐次スキャンは独立性レベル 1 または 2 のコミットされていない削除ローをブロックします。このオプションは、独立性レベル 1 と 2 のサーバ動作を変更します。

ほとんどの場合、このオプションは On のままにしてください。Off に設定すると、(使用可能なインデックスがある場合) ブロック動作はオプティマイザが選択したプランに左右されます。

recovery_time オプション [データベース]

データベース・サーバがシステム障害から回復するのにかかる最長時間を分で設定します。

指定可能な値

整数 (分)

デフォルト

2

スコープ

PUBLIC グループのみに設定できます。DBA 権限が必要です。サーバ再起動時に有効になります。

備考

このオプションは、いつチェックポイントを実行すべきかを決定するために、checkpoint_time オプションと一緒に使用します。

SQL Anywhere はヒューリスティックを使用して、最後のチェックポイント以後に行った操作に基づいてリカバリ時間を予測します。この予測には、データベースのリカバリ予想時間とチェックポイント予想時間も含まれます。そのため、リカバリ時間は正確ではありません。

参照

- 「自動リカバリ処理」 839 ページ
- 「checkpoint_time オプション [データベース]」 432 ページ
- 「-gr サーバ・オプション」 173 ページ

remote_idle_timeout オプション [データベース]

Web サービスのクライアント・プロシージャと関数で許容される休止時間 (秒数) を制御します。

指定可能な値

整数 (秒)

デフォルト

15

備考

このオプションは、Web サービスのクライアント・プロシージャと関数に影響します。アクティビティのない状態が指定の秒数を越えると、プロシージャまたは関数はタイムアウトになります。

replicate_all オプション [Replication Agent]

データベース全体が Replication Server 設定のプライマリ・サイトとして動作することを許可します。

指定可能な値

On、Off

デフォルト

Off

備考

このオプションは、SQL Anywhere Replication Agent でのみ使用されます。このオプションを On に設定すると、データベース全体が Replication Server インストール環境のプライマリ・サイトとして動作します。データベースに対する変更は、すべて Replication Agent によって Replication Server へ送信されます。

参照

- 「データベース全体のレプリケーション」 1024 ページ

replication_error オプション [SQL Remote]

SQL エラーが発生したときに Message Agent で呼び出すストアド・プロシージャを指定できます。

指定可能な値

ストアド・プロシージャ名

デフォルト

プロシージャなし

備考

SQL Remote で replication_error オプションを使用すると、SQL エラーが生じたときに Message Agent で呼び出すストアド・プロシージャを指定できるようにします。デフォルトではプロシージャを呼び出しません。

プロシージャには、データ型が CHAR、VARCHAR、または LONG VARCHAR の引数を 1 つ指定してください。プロシージャは、SQL エラー・メッセージで 1 回、エラーの原因となった SQL 文でもう 1 回呼び出されます。状況によっては (外部キー違反の場合など)、エラーの原因となった SQL 文が使用できないために、ストアド・プロシージャを 1 回しか呼び出すことができない場合があります。

このオプションを使用すると、レプリケーションで SQL エラーの追跡とモニタができますが、その場合でもやはり、エラーが起こらないように設計することが必要です。このオプションは、そのようなエラーを解決するためのものではありません。

参照

- 「SQL Remote オプション」 『SQL Remote』
- 「replication_error_piece オプション [SQL Remote]」 490 ページ

replication_error_piece オプション [SQL Remote]

replication_error オプションとともに使用することにより、SQL Remote レプリケーション中に SQL エラーが発生した場合に Message Agent によって呼び出される LONG VARCHAR ストアド・プロシージャを指定できます。

指定可能な値

ストアド・プロシージャ名

デフォルト

プロシージャなし

備考

エラーが発生し、replication_error が定義されている場合は、完全なエラー文字列とともに replication_error プロシージャが呼び出されます。

replication_error と replication_error_piece の両方が定義されている場合、エラーは VARCHAR 部に分割されます。replication_error が最初の VARCHAR 部とともに呼び出され、replication_error_piece が残りの VARCHAR 部とともに繰り返し呼び出されます。

参照

- 「replication_error オプション [SQL Remote]」 489 ページ
- 「SQL Remote オプション」 『SQL Remote』

request_timeout オプション [データベース]

1 つの要求を実行できる最大時間を設定します。このオプションを使用すると、接続が大量のサーバ・リソースを長時間にわたり消費することを防止できます。

指定可能な値

整数 (秒)、範囲は 0 ー 86400 (1 日)

デフォルト

0

備考

このオプションを 0 に設定すると、要求はタイムアウトになりません。

要求の実行時間が request_timeout に設定された時間 (CPU 時間ではなく実際の時間) を超えた場合、その要求は中断され、ユーザにエラーが返されます。返されるエラーは、SQLE_REQUEST_TIMEOUT「タイムアウトになったため、要求が中断されました」です。要求

がブロックされた場合、blocking_timeout が 0 に設定されていると、要求は最大で request_timeout に指定された時間にわたりブロックされたままとなり、その時間の経過後、ブロッキング・エラー (SQLE_LOCKED「'%2' のローは、ユーザ '%1' によってロックされています」など) を返します。

USER 値と PUBLIC 値 1 ー 14 は、使用できません。これによって、接続に時間がかかる場合でも (ログイン・プロシージャが複雑であるなどの理由で)、ユーザがサーバに接続できなくなることはありません。

このオプションは、データベース・クライアント要求と HTTP/HTTPS 要求のいずれに対しても使用できます。ただし、このオプションをストアド・プロシージャや HTTP/HTTPS 要求に設定しても、現在の要求には影響しません。要求の先頭にあるオプション値が使用されるからです。

request_timeout パブリック・オプションを設定する場合は注意が必要です。このオプションを設定すると、実行時間の長い要求を発行するアプリケーション (dbvalid、dbbackup、dbunload など) が失敗する可能性があります。また、大量のサーバ・リソースは使用しなくても別のユーザでブロックされる可能性のあるアプリケーションも、request_timeout が設定されていると失敗することがあります。このような問題に対処するには、ログイン・プロシージャでは接続の APPINFO 値に基づいて特定のアプリケーションだけに request_timeout オプションを設定します。

多数のローを含む結果セットをフェッチするなど、個々の要求を高速で評価するアプリケーションでは、このオプションを設定しても大量のサーバ・リソースの消費を回避できないことがあります。

参照

- 「blocking_timeout オプション [データベース]」 431 ページ
- 「AppInfo 接続パラメータ [APP]」 231 ページ

return_date_time_as_string オプション [データベース]

クエリが実行されたときに、日付、時刻、またはタイムスタンプの値がクライアント・アプリケーションに渡される方法を制御します。

指定可能な値

On、Off

デフォルト

Off

スコープ

現在の接続の間、テンポラリ・オプションとしてのみ設定できます。

備考

このオプションは、日付、時刻、タイムスタンプの値をアプリケーションに返すときに、日付または時刻データ型とするか文字列とするかを指定します。

このオプションを On に設定すると、timestamp_format、date_format、または time_format オプションの設定を保持するために、サーバは日付、時刻、またはタイムスタンプの値を文字列に変換してからクライアントに送信します。

Sybase Central と Interactive SQL では、return_date_time_as_string オプションは自動的に On に設定されます。

参照

- 「date_format オプション [互換性]」 441 ページ
- 「time_format オプション [互換性]」 504 ページ
- 「timestamp_format オプション [互換性]」 505 ページ

ri_trigger_time オプション [互換性]

参照整合性チェックとトリガ動作の相対タイミングを制御します。

指定可能な値

Before、After

デフォルト

After

スコープ

PUBLIC グループのみに設定できます。DBA 権限が必要です。

備考

このオプションは、Before または After に設定できます。After に設定すると、参照整合性アクションは UPDATE または DELETE の後に実行されます。

PUBLIC 設定だけが使用できます。他の設定は無視されます。

rollback_on_deadlock [データベース]

デッドロック発生時のトランザクションの処理方法を制御します。

指定可能な値

On、Off

デフォルト

On

スコープ

任意のユーザによって設定可能で、PUBLIC グループ用のほか、個々の接続用にも設定できます。すぐに有効になります。

備考

このオプションを On に設定すると、デッドロックが発生した場合、トランザクションは自動的にロールバックされます。ロールバックは、現在の要求が完了した後で発生します。このオプションを Off に設定した場合、SQL Anywhere はデッドロックが発生した文を自動的にロールバックし、そのトランザクションに対して、発生したデッドロックの種類を示すエラー・メッセージを返します。文のロールバックでは、その文によって取得されたロックが解放される可能性はあまりありません。

デッドロックの詳細については、「デッドロック」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

row_counts オプション [データベース]

クエリを開くときに、データベースがローの数をカウントするかどうかを指定します。

指定可能な値

On、Off

デフォルト

Off

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

このオプションを Off に設定すると、通常、ローの数は推定値のみとなります。On に設定すると、ローの正確な数がカウントされます。

> **警告**
> row_counts を On に設定すると、クエリの実行時間がかなり長くなることがあります。On に設定した場合、SQL Anywhere はクエリを 2 回実行するので、実行時間が倍になります。

save_remote_passwords オプション [SQL Remote]

メッセージ・リンクに入力されたパスワードを保存します。

指定可能な値

On、Off

デフォルト

On

備考

メッセージ・リンク・パラメータをデータベースではなく外部に保存する場合、パスワードが保存されないように設定できます。このオプションを Off に設定すると、パスワードは保存されません。

参照

- 「SQL Remote オプション」 『SQL Remote』

scale オプション [データベース]

計算結果が最大 precision にトランケートされる場合の、小数点以下の最大桁数を指定します。

指定可能な値

整数 (0 ー 127 の範囲で、かつ precision データベース・オプションの値より小さいことが必要)

デフォルト

6

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

掛け算、割り算、足し算、引き算、集合関数はすべて結果が最大精度を超えることができます。

参照

- 「precision オプション [データベース]」 483 ページ

secure_feature_key [データベース]

データベース・サーバの -sf オプションによって保護された機能を接続において有効にできます。

指定可能な値

文字列

デフォルト

NULL

スコープ

現在の接続の間、テンポラリ・オプションとしてのみ設定できます。

備考

データベース・サーバを起動するときに -sf オプションを指定することにより、サーバ上で動作するデータベースが使用できない機能を指定できます。-sk サーバ・オプションを使用すると、接続に対して保護されている (無効になっている) すべての機能を再び有効にするためのキーを

指定できます。また、データベース・サーバ上で実行されているすべてのデータベースに対して保護する機能を変更するための接続権限を付与します。データベース・サーバの起動時に、secure_feature_key テンポラリ・オプションの値を -sk オプションで指定されている値に設定すると、そのデータベース接続に対するすべての機能が再び有効化され、その接続では sa_server_option システム・プロシージャを使用してデータベース機能へのアクセスを制御できます。

secure_feature_key オプションを -sk で指定された値以外の値に設定すると、エラーは発生せず、-sf で指定された機能はその接続において無効なままとなります。

参照

- 「-sk サーバ・オプション」 195 ページ
- 「-sf サーバ・オプション」 192 ページ
- 「sa_server_option システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
- 「保護された機能の指定」 928 ページ

例

次のコマンドは、要求ログへのアクセスとすべてのリモート・データ・アクセス機能を無効にしてデータベース・サーバ secure_server を起動します。これらの機能は、-sk で指定したキーを使用して、特定のデータベース接続において有効にできます。

```
dbsrv10 -n secure_server -sf request_log,remote -sk j978kls12 testdb.db
```

secure_server データベース・サーバ上のデータベースに接続するときに secure_feature_key オプションを -sk で指定された値に設定すると、その接続において、要求ログへのアクセスとリモート・データ・アクセス機能が有効になります。

```
SET TEMPORARY OPTION secure_feature_key = 'j978kls12';
```

sort_collation オプション [データベース]

ORDER BY 式に対して SORTKEY 関数の暗黙の使用を許可します。

指定可能な値

Internal、collation_name、collation_id

デフォルト

Internal

備考

このオプションの値が Internal である場合、ORDER BY 句は変わりません。

このオプションの値を、有効な照合名または照合 ID に設定すると、ORDER BY 句の文字列式は、SORTKEY 関数が呼び出されたものとして扱われます。

参照

- 「SORTKEY 関数 [文字列]」 『SQL Anywhere サーバ - SQL リファレンス』

例

ソート照合をバイナリに設定します。

```
SET TEMPORARY OPTION sort_collation='binary';
```

ソート照合をバイナリに設定すると、以下のクエリに対して次のような変換が行われます。

```
SELECT Name, ID
FROM Products
ORDER BY Name, ID;
SELECT name, ID
FROM Products
ORDER BY 1, 2;
```

クエリは以下のように変換されます。

```
SELECT Name, ID
FROM Products
ORDER BY SORTKEY(Name, 'binary'), ID;
```

sql_flagger_error_level オプション [互換性]

指定された規格の一部ではない SQL への応答を制御します。

指定可能な値

- Off
- SQL:1992/Entry
- SQL:1992/Intermediate
- SQL:1992/Full
- SQL:1999/Core
- SQL:1999/Package
- SQL:2003/Core
- SQL:2003/Package
- Ultralite

デフォルト

Off

備考

このオプションは、指定された規格の一部ではない SQL をエラーとして通知します。たとえば、SQL:2003/Package を指定すると、データベース・サーバは、上級レベルの SQL/2003 構文ではない構文として通知します。

デフォルトの動作 (Off) では、エラー通知はオフに設定されます。

SQL Anywhere の以前のバージョンとの互換性を持たせるため、次の値も受け入れられ、次に示すようにマップされます。

- **E** このオプションは、SQL:1992/Entry に対応します。
- **I** このオプションは、SQL:1992/Intermediate に対応します。

- **F**　このオプションは、SQL:1992/Full に対応します。
- **W**　このオプションは、Off に対応します。

参照

- 「sa_ansi_standard_packages システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』
- 「SQLFLAGGER 関数 [その他]」『SQL Anywhere サーバ - SQL リファレンス』
- 「sql_flagger_warning_level オプション [互換性]」 497 ページ
- 「SQL プリプロセッサ」『SQL Anywhere サーバ - プログラミング』

sql_flagger_warning_level オプション [互換性]

指定された規格の一部ではない SQL への応答を制御します。

指定可能な値

- Off
- SQL:1992/Entry
- SQL:1992/Intermediate
- SQL:1992/Full
- SQL:1999/Core
- SQL:1999/Package
- SQL:2003/Core
- SQL:2003/Package
- Ultralite

デフォルト

Off

備考

このオプションは、指定された規格の一部ではない SQL を警告として通知します。たとえば、SQL:2003/Package を指定すると、データベース・サーバは、上級レベルの SQL/2003 構文ではない構文として通知します。

デフォルトの動作 (Off) では、警告通知はオフに設定されます。

以前のバージョンとの互換性を持たせるため、次の値も受け入れられ、次に示すようにマップされます。

- **E**　このオプションは、SQL:1992/Entry に対応します。
- **I**　このオプションは、SQL:1992/Intermediate に対応します。
- **F**　このオプションは、SQL:1992/Full に対応します。
- **W**　このオプションは、Off に対応します。

参照

- 「sa_ansi_standard_packages システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
- 「SQLFLAGGER 関数 [その他]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「sql_flagger_error_level オプション [互換性]」 496 ページ
- 「SQL プリプロセッサ」 『SQL Anywhere サーバ - プログラミング』

sr_date_format オプション [SQL Remote]

データベースから取り出した日付のフォーマットを設定します。

指定可能な値

文字列 (下記の記号の組み合わせ)

デフォルト

YYYY/*MM*/*DD*

備考

日付を保存するカラムをレプリケートするときに Message Agent が使用します。フォーマットは次の記号を組み合わせた文字列です。

シンボル	説明
YY	2 桁の年
YYYY	4 桁の年
MM	2 桁の月
MMM[*m*…]	月を略した文字、"m" の数だけ文字を使用
DD	2 桁の日

各記号は、フォーマットされる日付のデータで置き換えられます。

文字データを表す記号 (*MMM* など) では、出力の文字を次のように制御できます。

- 記号をすべて大文字で入力すると、フォーマットはすべて大文字で表記されます。たとえば、MMM と入力すると、JAN と表記されます。
- 記号をすべて小文字で入力すると、フォーマットはすべて小文字で表記されます。たとえば、mmm と入力すると、jan と表記されます。
- 大文字と小文字を混ぜて入力すると、使用される言語に適切な文字が SQL Anywhere により選択されます。たとえば、Mmm と入力すると、英語では May、フランス語では Mai と表記されます。

文字データがマルチバイトである場合、各記号の長さはバイト数ではなく文字数を表します。たとえば、'mmm' は 3 文字の月名を示しています。

数値データを表す記号では、記号に大文字を使用するか小文字を使用するかで 0 埋め込みを制御できます。

このオプションは、次の記号で構成される文字列です。

- 記号をすべて大文字または小文字 (MM や mm など) で入力すると、0 埋め込みが行われます。たとえば、yyyy/mm/dd と入力すると、2002/01/01 と表記されます。
- 大文字と小文字を混ぜて入力すると (Mm など)、0 の埋め込みは行われません。たとえば、yyyy/Mm/Dd と入力すると、2002/1/1 と表記されます。

参照

- 「sr_time_format オプション [SQL Remote]」 499 ページ
- 「sr_timestamp_format [SQL Remote]」 500 ページ
- 「SQL Remote オプション」 『SQL Remote』

sr_time_format オプション [SQL Remote]

データベースから取り出した時刻のフォーマットを設定します。

指定可能な値

文字列 (下記の記号の組み合わせ)

デフォルト

HH:*NN*:*SS*.*SSSSS*

備考

時間を保存するカラムをレプリケートするときに Message Agent が使用します。フォーマットは次の記号を組み合わせた文字列です。

シンボル	説明
HH	2 桁の時間 (24 時間表示)。
NN	2 桁の分。
MM	コロンの後の場合は、2 桁の分 (hh:mm など)。
SS.*ssssss*	秒と小数点第 6 位までの秒の小数。小数点以下 6 桁のタイムスタンプをサポートしないプラットフォームもあります。

各記号は、フォーマットしようとする日付のデータで置き換えられます。数字の出力ではなく文字を表すフォーマット記号は、大文字で入力でき、その場合、置き換えられる文字も大文字になります。フォーマット文字列で大文字と小文字を混ぜて使用すると、数字の前にゼロが付きません。

備考

フォーマット文字列で大文字と小文字を混在させて使用すると、数字の前にゼロが付きません。

参照

- 「sr_date_format オプション [SQL Remote]」 498 ページ
- 「sr_timestamp_format [SQL Remote]」 500 ページ
- 「SQL Remote オプション」 『SQL Remote』

sr_timestamp_format [SQL Remote]

データベースから取り出したタイムスタンプのフォーマットを設定します。

指定可能な値

フォーマット文字列は sr_timestamp_format オプション と sr_timestamp_format オプション設定から取得します。

デフォルト

yyyy/*mm*/*dd hh*:*nn*:*ss*.*Ssssss*

備考

このオプションは、Message Agent で日付情報をレプリケートするときに使用します。デフォルト設定は、sr_date_format オプション設定と sr_time_format オプション設定を組み合わせたものです。

参照

- 「sr_date_format オプション [SQL Remote]」 498 ページ
- 「sr_time_format オプション [SQL Remote]」 499 ページ
- 「SQL Remote オプション」 『SQL Remote』

string_rtruncation オプション [互換性]

INSERT か UPDATE が CHAR または VARCHAR 文字列をトランケートするときに、エラーを出すかどうかを決定します。

指定可能な値

On、Off

デフォルト

On

備考

トランケート文字がスペースだけで構成されている場合、例外は発生しません。On に設定すると、ANSI/ISO SQL/2003 の動作に対応します。Off に設定すると、例外は発生せず、文字列は何も通知されずにトランケートされます。

参照

- 「文字データ型」 『SQL Anywhere サーバ - SQL リファレンス』

subscribe_by_remote オプション [SQL Remote]

SUBSCRIBE BY 値が NULL または空の文字列である場合の解釈を制御します。

指定可能な値

On、Off

デフォルト

On

備考

このオプションを On に設定すると、NULL または空の文字列である SUBSCRIBE BY 値のあるローに対してリモート・データベースから操作が行われた場合、リモート・ユーザがローにサブスクリプションを作成したと見なします。Off に設定すると、リモート・ユーザはローにサブスクリプションを作成していないと見なされます。

このオプションでの唯一の制約は、リモート・ユーザが NULL 値のローまたは空のサブスクリプション式 (情報が統合データベースにだけ格納されているもの) を、実際に INSERT (または UPDATE) しようとした場合にエラーが発生することです。この問題はそれほど深刻ではなく、インストール環境で、リモート・ユーザに属さないサブスクリプション値を割り当てることで対処できます。

参照

- 「SQL Remote オプション」 『SQL Remote』
- 「多対多の関係における subscribe_by_remote オプションの使用」 『SQL Remote』

subsume_row_locks オプション [データベース]

サーバがテーブル用に個別のロー・ロックを獲得したときに制御します。

指定可能な値

On、Off

デフォルト

On

備考

subsume_row_locks オプションが On (デフォルト) の場合、LOCK TABLE *t* IN EXCLUSIVE MODE で排他的にテーブル *t* がロックされるたびに、サーバが *t* 用に個別のロー・ロックを取得しなくなります。

これにより、単一のトランザクションで広範な更新が *t* に対して実行された場合 (特に *t* がキャッシュ・サイズに比べて大きい場合)、パフォーマンスが大幅に改善されます。また、ロック・テーブルが現在扱えるよりも、より大きいアトミック更新オペレーションができるようになりました (約 200 万ー 400 万ロー)。

このオプションが On のときに、テーブル上のキーセット・カーソルがこのような形でロックされると、データベース内のいずれかのローが変更される場合にカーソル内のすべてのローに対してロー変更警告が返されます。サーバは ORDER BY のある更新可能なカーソルを、結果としてキーセット・カーソルにすることができることに注意してください。

suppress_tds_debugging オプション [データベース]

TDS デバッグ情報をサーバ・メッセージ・ウィンドウに表示するかどうかを決定します。

指定可能な値

On、Off

デフォルト

Off

備考

サーバを -z オプションで起動すると、TDS プロトコルに関するデバッグ情報を含むデバッグ情報がサーバ・メッセージ・ウィンドウに表示されます。

suppress_tds_debugging オプションは、TDS に関するデバッグ情報がサーバ・メッセージ・ウィンドウに表示されないようにします。このオプションを Off (デフォルト) に設定すると、TDS デバッグ情報がサーバ・メッセージ・ウィンドウに表示されます。

synchronize_mirror_on_commit オプション [データベース]

非同期モードまたは非同期フルページ・モードにおいて、データベースの変更がミラー・サーバへ送信されたことがどの時点で保証されるかを制御します。

指定可能な値

On、Off

デフォルト

Off

備考

synchronize_mirror_on_commit オプションでは、非同期モードまたは非同期フルページ・モードにおいて、データベースの変更がミラー・サーバへ送信されたことがどの時点で保証されるかについて厳密に制御できます。このオプションのデフォルトは Off です。On に設定すると、COMMIT が発生するたびにトランザクション・ログに記録されたすべての変更がミラー・サーバへ送信され、それらの変更がミラー・サーバで受信されると、ミラー・サーバからプライマリ・サーバへ受信確認が送信されます。このオプションは、SET TEMPORARY OPTION を使用して特定のトランザクションに対して設定できます。また、ログイン・プロシージャで APPINFO 文字列を調べ、このオプションを特定のアプリケーションだけに設定すると、効果的な場合もあります。これによって、ミラーリング動作をさまざまなアプリケーションのニーズに適合させることができます。

参照

- 「データベース・ミラーリングの概要」 888 ページ

tds_empty_string_is_null オプション [データベース]

TDS 接続で空の文字を NULL として返すか、ブランク文字 1 文字を含む文字列として返すかを制御します。

指定可能な値

On、Off

デフォルト

Off

備考

デフォルトでは、このオプションには Off が設定され、空の文字列は、TDS 接続用の 1 文字のブランク文字を含む文字列で返されます。このオプションを On に設定すると、空の文字列は、TDS 接続では NULL 文字列として返されます。TDS 以外の接続では、空の文字列と NULL 文字列は区別されます。

temp_space_limit_check オプション [データベース]

接続によって使用されるテンポラリ・ファイル領域のサイズを確認し、要求された領域サイズが接続に許容されるサイズよりも大きい場合は要求は失敗になります。

指定可能な値

On、Off

デフォルト

On

スコープ

PUBLIC グループのみに設定できます。DBA 権限が必要です。

備考

temp_space_limit_check オプションを On に設定すると、接続がそのクォータを超えるテンポラリ・ファイル領域を要求した場合、その要求は失敗し、エラー SQLSTATE_TEMP_SPACE_LIMIT が返されます。このオプションを Off (デフォルト) に設定すると、データベース・サーバは接続が使用するテンポラリ・ファイル領域のサイズをチェックしません。接続がそのクォータを超えるテンポラリ領域を要求した場合、このオプションが Off に設定されていると、致命的なエラーが発生することがあります。

次の 2 つのしきい値のうち小さい方が、接続のテンポラリ・ファイル領域のクォータになります。

1. max_temp_space オプションで指定された各接続に許可されるテンポラリ・ファイル領域の最大サイズ
2. テンポラリ・ファイルのサイズを接続数で除算した見込みの最大サイズ

しきい値は、テンポラリ・ファイルのサイズが最大サイズの 80% に達した場合にのみ使用されます。これは、オペレーティング・システムによって報告される、デバイス上に残っている空き領域のサイズに基づいて判断されます。接続要求によって、クォータが許可するサイズよりも大きいテンポラリ・ファイル領域が要求された場合、接続の現在の要求は SQLSTATE 54W05 (TEMP_SPACE_LIMIT) で失敗します。

接続が使用するテンポラリ・ファイル領域の最大サイズは、max_temp_space オプションで指定できます。

参照

- 「sa_disk_free_space システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
- 「max_temp_space オプション [データベース]」 469 ページ

time_format オプション [互換性]

データベースから取り出した時刻のフォーマットを設定します。

指定可能な値

文字列 (下記の記号の組み合わせ)

デフォルト

HH:*NN*:*SS.SSS*

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

フォーマットは次の記号を組み合わせた文字列です。

シンボル	説明
HH	2 桁の時間
NN	2 桁の分
SS.ssssss	秒と小数点第 6 位までの秒の小数。小数点以下 6 桁のタイムスタンプをサポートしないプラットフォームもあります。
AA	A.M. または P.M. (12 時間表記)–24 時間表記の場合、*AA* と *PP* は省略します。
PP	PM. (必要に応じて) (12 時間表記)–24 時間表記の場合、*AA* と *PP* は省略します。

各記号は、フォーマットしようとする日付のデータで置き換えられます。数字の出力ではなく文字を表すフォーマット記号は、大文字で入力でき、その場合、置き換えられる文字も大文字になります。フォーマット文字列で大文字と小文字を混ぜて使用すると、数字の前にゼロが付きません。

参照

- 「date_format オプション [互換性]」 441 ページ
- 「timestamp_format オプション [互換性]」 505 ページ

time_zone_adjustment オプション [データベース]

接続のタイム・ゾーン調整を修正できます。

指定可能な値

整数 (例： 300)

引用符で囲んだ負の整数 (例： '-300')

先頭が + または - で、引用符で囲まれた時と分を表す文字列 (例： '+5:00'、'-5:00')

デフォルト

クライアントが Embedded SQL、ODBC、OLE DB、ADO、または ADO.NET を介して接続している場合は、クライアントのタイム・ゾーンに従ってデフォルト値が設定されます。クライアントが jConnect または Open Client を介して接続している場合は、デフォルトは、サーバのタイム・ゾーンに基づいて設定されます。

備考

time_zone_adjustment オプションの値は SELECT CONNECTION_PROPERTY ('TimeZoneAdjustment'); が返す値と同じです。この値は、接続のローカル時間を表示するために協定世界時 (UTC: Coordinated Universal Time) に加算する必要がある分数を表します。

参照

- 「接続レベルのプロパティ」 518 ページ

timestamp_format オプション [互換性]

データベースから取り出したタイムスタンプのフォーマットを設定します。

指定可能な値

文字列 (下記の記号の組み合わせ)

デフォルト

YYYY-MM-DD HH:NN:SS.SSS

Open Client 接続と JDBC 接続の場合、デフォルトでは *YYYY-MM-DD HH:NN:SS.SSS* に設定されます。

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

フォーマットは次の記号を組み合わせた文字列です。

シンボル	説明
YY	2 桁の年
YYYY	4 桁の年
MM	2 桁の月、またはコロンの後の場合は 2 桁の分 ('*HH:MM*' など)
MMM[*m*…]	月を略した文字、"m" の数だけ文字を使用
DD	2 桁の日
DDD[*d*…]	曜日を略した文字
HH	2 桁の時間
NN	2 桁の分
SS.ssssss	秒と小数点第 6 位までの秒の小数。小数点以下 6 桁のタイムスタンプをサポートしないプラットフォームもあります。
AA	A.M. または P.M. (12 時間表記)−24 時間表記の場合、*AA* と *PP* は省略します。
PP	PM. (必要に応じて) (12 時間表記)−24 時間表記の場合、*AA* と *PP* は省略します。

各記号は、フォーマットされる日付のデータで置き換えられます。

文字データを表す記号 (*MMM* など) では、出力の文字を次のように制御できます。

- 記号をすべて大文字で入力すると、フォーマットはすべて大文字で表記されます。たとえば、MMM と入力すると、JAN と表記されます。
- 記号をすべて小文字で入力すると、フォーマットはすべて小文字で表記されます。たとえば、mmm と入力すると、jan と表記されます。
- 大文字と小文字を混ぜて入力すると、使用される言語に適切な文字が SQL Anywhere により選択されます。たとえば、Mmm と入力すると、英語では May、フランス語では Mai と表記されます。

文字データがマルチバイトである場合、各記号の長さはバイト数ではなく文字数を表します。たとえば、'mmm' は 3 文字の月名を示しています。

数値データを表す記号では、記号に大文字を使用するか小文字を使用するかで 0 埋め込みを制御できます。

- 記号をすべて大文字または小文字 (MM や mm など) で入力すると、0 埋め込みが行われます。たとえば、yyyy/mm/dd と入力すると、2002/01/01 と表記されます。
- 大文字と小文字を混ぜて入力すると (Mm など)、0 の埋め込みは行われません。たとえば、yyyy/Mm/Dd と入力すると、2002/1/1 と表記されます。

参照

- 「date_format オプション [互換性]」 441 ページ
- 「time_format オプション [互換性]」 504 ページ

truncate_timestamp_values オプション [データベース] [Mobile Link クライアント]

タイムスタンプ値の精度を制限します。

指定可能な値

On、Off

デフォルト

Off

スコープ

PUBLIC グループのみに設定できます。DBA 権限が必要です。このオプションは、タイムスタンプ・データがすでに格納されているデータベースに対しては有効にしないでください。

備考

SQL Anywhere の TIMESTAMP 値の精度は、小数点以下 6 桁です。ただし、TIMESTAMP 値を小数点以下 3 桁などにトランケートする他のソフトウェアとの互換性を維持するために、truncate_timestamp_values オプションを On に設定すると、SQL Anywhere が格納する小数点以下の桁数を制限できます。default_timestamp_increment オプションは、TIMESTAMP 値を小数点以下何桁でトランケートするかを決定します。

Mobile Link 同期に対し、このオプションを設定する場合は、最初の同期の実行前に設定する必要があります。

データベース・サーバが truncate_timestamp_values と default_timestamp_increment の組み合わせで指定されたものより細かい TIMESTAMP 値を検出した場合、エラーがレポートされます。

ほとんどの場合、データベースをアンロードしてそれを truncate_timestamp_values 値と default_timestamp_increment 値が設定されている新しいデータベースに再ロードする方法が、確実に適切な TIMESTAMP 値を使用する一番簡単なソリューションです。ただし、テーブル内の TIMESTAMP カラムのタイプにより、次のように行うこともできます。

- TIMESTAMP カラムが DEFAULT TIMESTAMP または DEFAULT UTC TIMESTAMP で定義されている場合 (つまり、ローが変更されるとサーバによって値が自動的に更新される)、truncate_timestamp_values オプションを変更する前にテーブル内のすべてのローを削除する必要があります。ローは DELETE 文または TRUNCATE TABLE 文を使用して削除します。

- TIMESTAMP カラムが DEFAULT TIMESTAMP または DEFAULT UTC TIMESTAMP 以外で定義されている場合、UPDATE 文を実行して文字列に値を割り当てて、TIMESTAMP に戻します。次に例を示します。

```
UPDATE T
  SET ts = CAST( DATEFORMAT( ts, 'yyyy/mm/dd hh:nn:ss.ss')
AS TIMESTAMP);
```

この手順は必要とされる精度より低下する可能性があることに注意してください。使用するフォーマット文字列は、保持する精度の桁数によります。

参照

- 「default_timestamp_increment オプション [データベース] [Mobile Link クライアント]」 445 ページ

例

たとえば、default_timestamp_increment オプションを 100000 に設定すると、秒の部分は小数点以下 1 桁の後でトランケートされるので、"2000/12/05 10:50:53.700" のような値を格納できるようになります。

truncate_with_auto_commit オプション [データベース]

TRUNCATE TABLE 文の処理を高速化します。

指定可能な値

On、Off

デフォルト

On

スコープ

PUBLIC グループのみに設定できます。DBA 権限が必要です。

備考

truncate_with_auto_commit を On に設定すると、TRUNCATE TABLE 文の実行前と実行後に COMMIT が実行されます。このオプションの主な目的は、テーブルのトランケーション (すべてのローの削除) を高速に行うことです。

次のような場合は、高速トランケートが実行できません。

- テーブルへの外部キー、またはテーブルからの外部キーがある場合
- TRUNCATE TABLE 文がトリガ内で実行された場合
- TRUNCATE TABLE 文がアトミック文内部で実行された場合

参照

- 「TRUNCATE TABLE 文」 『SQL Anywhere サーバ - SQL リファレンス』

tsql_hex_constant オプション [互換性]

16 進定数がバイナリ文字定数として処理されるかどうかを制御します。

指定可能な値

On、Off

デフォルト

On

備考

このオプションを On に設定すると、16 進定数はバイナリ文字定数として処理されます。従来の動作にするには、オプションを Off に設定します。

tsql_outer_joins オプション [互換性]

Transact-SQL 外部ジョイン演算子の *= と =* を文やビューで使用可能にするかどうかを指定します。

指定可能な値

On、Off

デフォルト

Off

備考

Transact-SQL 外部ジョインはサポートされなくなりました。このオプションを On に設定すると、Transact-SQL 外部ジョインを使用できます。

tsql_variables オプション [互換性]

@ 符号を、Embedded SQL ホスト変数名のプレフィクスとして使用できるかどうかを制御します。

指定可能な値

On、Off

デフォルト

Off

Open Client 接続と JDBC 接続の場合は On

備考

このオプションを On に設定すると、Embedded SQL のホスト変数名のプレフィクスとしてコロンの代わりに @ 符号を使用できます。これは、主に Transact-SQL との互換性を保つために実装されています。

updatable_statement_isolation オプション [データベース]

isolation_level オプションが readonly-statement-snapshot に設定されている場合に更新可能な文に適用する独立性レベルを指定します。

指定可能な値

0, 1, 2, 3

デフォルト

0

備考

updatable_statement_isolation オプションで指定した独立性レベルは、isolation_level オプションが readonly-statement-snapshot に設定されている場合に、更新可能な文によって使用されます。次の値を指定できます。

- **0** ダーティ・リード、繰り返し不可能読み出し、幻ローを許可します。
- **1** ダーティ・リードを防ぎます。繰り返し不可能読み出しと幻ローを許可します。
- **2** ダーティ・リードと繰り返し不可能読み出しを防ぎます。幻ローを許可します。
- **3** 直列化可能。ダーティ・リード、繰り返し不可能読み出し、幻ローを防ぎます。

参照

- 「isolation_level オプション [互換性]」 455 ページ
- 「スナップショット・アイソレーション」 『SQL Anywhere サーバ - SQL の使用法』
- 「独立性レベルと一貫性」 『SQL Anywhere サーバ - SQL の使用法』
- 「独立性レベルの選択」 『SQL Anywhere サーバ - SQL の使用法』

update_statistics オプション [データベース]

クエリ実行中の統計上の収集を制御します。

指定可能な値

On、Off

デフォルト

On

備考

サーバは通常のクエリ実行中に統計情報を収集し、収集した統計に使用してカラム統計を自己チューニングします。update_statistics オプションを Off に設定すると、クエリ実行中の統計情報の収集を無効にできます。

通常の環境において、このオプションをオフにする必要はありません。

update_statistics オプションは、データの更新 (LOAD/INSERT/UPDATE/DELETE) による統計の変更に影響しません。これらの文に基づいて統計を更新するかどうかを設定するには、collect_statistics_on_dml_updates データベース・オプションを使用します。

collect_statistics_on_dml_updates オプションと update_statistics オプションが異なる点は、update_statistics オプションを On に設定した場合、ある述部を満たすローの実際の数とその述部を満たすと予想されるローの数が比較され、その結果に基づいて推定値が更新されることです。collect_statistics_on_dml_updates オプションを On に設定した場合は、挿入、更新、または削除されたローの値に基づいてカラム統計が修正されます。

参照

- 「collect_statistics_on_dml_updates オプション [データベース]」 434 ページ
- 「カラム統計の更新」 『SQL Anywhere サーバ - SQL の使用法』

user_estimates オプション [データベース]

クエリの述部に含まれるユーザ選択性推定をクエリ・オプティマイザが尊重するか無視するかを制御します。

指定可能な値

Enabled、Disabled、Override-Magic

デフォルト

Override-Magic

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

SQL Anywhere では、ユーザ選択性推定を指定することで、サーバが述部の選択性を正確に推定できないときにオプティマイザのパフォーマンスを向上させることができます。ただし、ユーザ選択性推定は、適切な状況でのみ使用してください。たとえば、オプティマイザが使用する Override-Magic 選択性推定と実際の選択性が大きく異なっている場合は、1 つ以上の関数が関係する述部に対して選択性推定を指定することは有用です。

ソフトウェアによって選択されたアクセス・プランが不適切であり、パフォーマンス問題を回避するために選択性推定を使用したものの、それが不正確であった場合は、このオプションを Disabled に設定することをおすすめします。不正確な推定が使用された場合は、サーバは最適なプランを選択できません。

ユーザ選択性推定の詳細については、「明示的な選択性推定」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

ユーザ選択性推定に述部が指定されているときは、このオプションの設定に基づいて、その推定は尊重されるか無視されます。次の値を指定できます。

- **Enabled** ユーザが提供する選択性推定をすべて尊重します。このオプションは、On を使用して有効にすることもできます。
- **Override-Magic** ユーザ選択性推定は尊重されますが、オプティマイザが最後の手段であるヒューリスティック値 (マジック値とも呼ばれます) の使用を選択する以外に方法がない場合のみ使用されます。
- **Disabled** 他の推定データが使用できないときは、ユーザ推定は無視され、マジック値が使用されます。このオプションは、Off を使用して無効にすることもできます。

uuid_has_hyphens オプション [データベース]

ユニークな識別子の値が文字列に変換するときのフォーマットを設定します。

指定可能な値

On、Off

デフォルト

On

備考

uid_has_hyphens オプションを On に設定すると、変換後の文字列には 4 つのハイフンが挿入されます。たとえば、12345678-1234-1234-1234-1234567890ab のようになります。このオプションを Off に設定すると、文字列にハイフンは挿入されません。文字列からユニークな識別子の値への変換では、データベース・サーバはハイフンが挿入された UUID 文字列とハイフンのない UUID 文字列の両方をサポートします。

参照

- 「UNIQUEIDENTIFIER データ型」『SQL Anywhere サーバ - SQL リファレンス』

verify_all_columns オプション [SQL Remote]

ローカル・データベースで発行した更新を含むメッセージがすべてのカラム値を含んで送信されるかどうかを制御します。

指定可能な値

On、Off

デフォルト

Off

備考

このオプションは、SQL Remote のみで使用します。On に設定すると、ローカル・データベースの発行した更新を含むメッセージはすべてのカラム値を含んで送信され、カラムの重複はサブスクライバ・データベースで RESOLVE UPDATE トリガを起動します。

例

次の文は、SQL Anywhere で、すべてのユーザに対して verify_all_columns オプションを Off に設定します。

```
SET OPTION PUBLIC.verify_all_columns = 'Off'
```

参照

- 「SQL Remote オプション」 『SQL Remote』

verify_password_function オプション [データベース]

パスワード規則 (1 桁以上の長さであることなど) の実装に使用できる関数を指定します。この関数は GRANT CONNECT TO *userid* IDENTIFIED BY *password* 文で呼び出されます。

指定可能な値

文字列

デフォルト

空の文字列 (GRANT CONNECT 文で関数が呼び出されない)

スコープ

DBA 権限が必要です。

備考

verify_password_function オプションの値を空の文字列以外に設定すると、指定した関数が GRANT CONNECT TO *userid* IDENTIFIED BY *password* 文によって呼び出されます。このオプションには、*owner.function_name* という形式の値を指定する必要があります。これによって、ユーザが関数を無効にすることを防止できます。SQL Anywhere では、パスワードの大文字と小文字が区別されます。

GRANT CONNECT 文が有効であることが確認されると (ユーザに付与を実行するパーミッションがあるなど)、このオプションで指定した関数が呼び出され、関数に指定された規則に基づいてパスワードが検証されます。パスワードが規則に従っている場合は、関数が成功を示す NULL を返し、付与が実行されます。パスワードが規則に違反している場合は、エラーが設定されるか、エラーを示す NULL 以外の文字列が返されます。NULL 以外の文字列は、パスワードが不合格となった理由として、ユーザに返されるエラーに追加されます。

パスワード検証関数は、*user_name* VARCHAR(128) と *new_pwd* VARCHAR(255) という 2 つのパラメータを取ります。戻り値のデータ型は VARCHAR(255) です。パスワード検証関数がデバッガによってステップスルーされないように、この関数で ALTER FUNCTION *function-name* SET HIDDEN 文を実行することをおすすめします。verify_password_function オプションを設定する場合、GRANT CONNECT で複数のユーザ ID とパスワードを指定することはできません。

パスワード再利用の禁止やパスワード有効期限の運用などの詳細なパスワード規則を扱った例を含め、パスワード規則の詳細については、「パスワード検証関数の使用」 923 ページを参照してください。

参照

- 「min_password_length オプション [データベース]」 470 ページ
- 「GRANT 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「ALTER FUNCTION 文」『SQL Anywhere サーバ - SQL リファレンス』

例

次の例は、パスワードがユーザのユーザ ID と同じであるかどうかをチェックする f_verify_pwd 関数を作成します。パスワードとユーザ ID と同じである場合、ユーザは別のパスワードを指定する必要があります。

```
CREATE FUNCTION DBA.f_verify_pwd( user_name VARCHAR(128),
                    new_pwd VARCHAR(255) )
RETURNS VARCHAR(255)
BEGIN
   -- enforce password rules
   IF new_pwd = user_name THEN
      RETURN( 'password cannot be the same as the user name' );
   END IF;
   -- return success
   RETURN( NULL );
END;

ALTER FUNCTION DBA.f_verify_pwd SET HIDDEN;
GRANT EXECUTE On DBA.f_verify_pwd TO PUBLIC;
SET OPTION PUBLIC.verify_password_function = 'DBA.f_verify_pwd';
```

verify_threshold オプション [SQL Remote]

更新がレプリケートされるときにどのカラムが確認されるかを制御します。

指定可能な値

整数 (バイト)

デフォルト

1000

備考

このオプションは、SQL Remote のみで使用します。カラムのデータ型がしきい値より長い場合は、カラムの古い値は UPDATE がレプリケートされるときに確認されません。このようにして、SQL Remote メッセージのサイズが大きくならないようにしますが、競合する長い値の更新が検出されないという短所もあります。

参照

- 「SQL Remote オプション」『SQL Remote』

wait_for_commit オプション [データベース]

データが操作されるときに、いつ外部キー整合性をチェックするかを決定します。

指定可能な値

On、Off

デフォルト

Off

スコープ

個々の接続または PUBLIC グループに設定できます。すぐに有効になります。

備考

このオプションを On に設定すると、データベースは次の COMMIT 文まで外部キー整合性をチェックしません。Off の場合は、check_on_commit オプションで作成されていないすべての外部キーが、挿入、更新、削除時にチェックされます。

webservice_namespace_host オプション [データベース]

生成された WSDL ドキュメント内で XML ネームスペースとして使用するホスト名を指定します。DBA 権限が必要です。

指定可能な値

NULL または hostname-string

デフォルト

NULL

スコープ

PUBLIC グループのみに設定できます。すぐに有効になります。このオプションを設定するには、DBA 権限が必要です。

備考

Webservices Description Language Documents (WSDL) は、DISH サービスによってエクスポートされます。WSDL は、使用可能な SOAP サービスの説明を含んだ XML ドキュメントです。XML ドキュメント内の targetNameSpace と soapAction 操作の URL にはホスト名が含まれます。このオプションがデフォルト値の NULL に設定されている場合、ホスト名はデータベース・サーバ実行中のコンピュータのホスト名になります。このオプションが文字列値に設定されていると、ホスト名にはその文字列が使用されます。このオプションは、配備後は開発に使用したホスト以外のホストを対象とする Web サービス・クライアント・アプリケーションを開発するときに使用することを目的としています。

第 13 章

データベース・プロパティ

目次

データベース・プロパティの概要

SQL Anywhere には、クライアント・アプリケーションで使用できる各種のプロパティが用意されています。これらのプロパティは、接続、データベース、データベース・サーバの動作を示します。プロパティ名には大文字と小文字の区別はありません。

プロパティへのアクセス

各プロパティへアクセスするには、プロパティ名をシステム関数の引数として指定します。

- **接続プロパティにアクセスするには、次の手順に従います。**

  - CONNECTION_PROPERTY システム関数を使用します。次の文は、現在の接続によってファイルから読み込まれたページの数を返します。

    ```
    SELECT CONNECTION_PROPERTY ( 'DiskRead' );
    ```

- **データベース・プロパティにアクセスするには、次の手順に従います。**

  - DB_PROPERTY システム関数を使用します。たとえば、次の文は、現在のデータベースのページ・サイズを返します。

    ```
    SELECT DB_PROPERTY ( 'PageSize' );
    ```

- **データベース・サーバ・プロパティにアクセスするには、次の手順に従います。**

  - PROPERTY システム関数を使用します。次の文は、グローバル・サーバ・データ構造に使用されたキャッシュ・ページの数を返します。

    ```
    SELECT PROPERTY ( 'MainHeapPages' );
    ```

接続レベルのプロパティ

次の表は、各接続で使用できるプロパティのリストです。

例

- **接続プロパティの値を取り出すには、次の手順に従います。**

  - CONNECTION_PROPERTY システム関数を使用します。次の文は、現在の接続によってファイルから読み込まれたページ数を返します。

    ```
    SELECT CONNECTION_PROPERTY ( 'DiskRead' );
    ```

- **すべての接続プロパティの値を取り出すには、次の手順に従います。**

  - sa_conn_properties システム・プロシージャを使用します。

    ```
    CALL sa_conn_properties;
    ```

    接続ごとに個別のローが表示されます。

参照

- 「サーバ・レベルのプロパティ」 540 ページ
- 「データベース・レベルのプロパティ」 550 ページ

説明

プロパティ	説明
allow_nulls_by_default	NULL と NOT NULL のいずれも指定せずに作成されたカラムについて NULL 値の入力を許可するかどうかを示す値を返します。「allow_nulls_by_default オプション [互換性]」 421 ページを参照してください。
allow_snapshot_isolation	スナップショット・アイソレーションが有効であるか無効であるかを示す値を返します。「allow_snapshot_isolation オプション [データベース]」 422 ページを参照してください。
ansi_blanks	どのような場合に文字データがクライアント側でトランケートされるかを示す値を返します。「ansi_blanks オプション [互換性]」 423 ページを参照してください。
ansi_close_cursors_on_rollback	WITH HOLD で開かれたカーソルが ROLLBACK の実行時に閉じるかどうかを示す値を返します。「ansi_close_cursors_on_rollback オプション [互換性]」 424 ページを参照してください。
ansi_integer_overflow	計算式が整数のオーバフロー・エラーを起こしたときに発生する事象を示す値を返します。「ansi_integer_overflow オプション [互換性]」 424 ページを参照してください。
ansi_permissions	DELETE 文と UPDATE 文についてパーミッションをチェックするかどうかを示す値を返します。「ansi_permissions オプション [互換性]」 424 ページを参照してください。
ansi_substring	start パラメータまたは length パラメータに負の値が設定された場合に SUBSTRING (SUBSTR) 関数がどのような動作をするかを示す値を返します。「ansi_substring オプション [互換性]」 425 ページを参照してください。
ansi_update_constraints	更新可能な範囲を示す値を返します。「ansi_update_constraints オプション [互換性]」 426 ページを参照してください。
ansinull	NULL をどのように解釈するかを示す値を返します。「ansinull オプション [互換性]」 427 ページを参照してください。

プロパティ	説明
AppInfo	接続を確立したクライアントに関する情報を返します。HTTP 接続では、ブラウザの情報が含まれています。jConnect または Open Client の古いバージョンを使った接続については、情報は不完全の場合があります。 API 値は、DBLIB、ODBC、OLEDB、ADO.NET、iAnywhereJDBC、PHP、PerlDBD、DBEXPRESS のいずれかです。 その他のタイプの接続について返される値の詳細については、「AppInfo 接続パラメータ [APP]」 231 ページを参照してください。
ApproximateCPUTime	特定の接続で使用された累積 CPU 時間の概算値 (秒単位) を返します。この値と実際の CPU 時間との誤差は、通常は 5ー10 % ですが、最大で 50 % に達することもあります。マルチプロセッサ・コンピュータでは、CPU (ハイパースレッドまたはコア) ごとに使用時間が積算されるため、すべての接続の累積時間を合計した値が実際の経過時間より大きくなる場合もあります。このプロパティは Windows と Linux でサポートされています。
auditing	PUBLIC.auditing オプションが On に設定されている場合は、On を返します。それ以外の場合は、Off を返します。 PUBLIC.auditing オプションが On に設定され、conn_auditing オプションが Off に設定されている場合は、現在の接続が監査されていなくても、auditing 接続プロパティは On を返します。「監査の有効化」 930 ページと「auditing オプション [データベース]」 428 ページを参照してください。
auditing_options	このプロパティはシステムで使用するために予約されています。このオプションの設定は変更しないでください。
automatic_timestamp	TIMESTAMP データ型の新規カラムがどのように解釈されるかを示す値を返します。「automatic_timestamp オプション [互換性]」 429 ページを参照してください。
background_priority	現在の接続が他の接続のパフォーマンスにもたらす影響の大きさを示す値を返します。「background_priority オプション [データベース]」 430 ページを参照してください。
BlockedOn	現在の接続がブロックされていない場合は 0 を返し、ブロックされている場合はロック競合によってブロックされた接続の接続番号を返します。
blocking	ロック競合に対するデータベース・サーバの動作を示す値を返します。「blocking オプション [データベース]」 431 ページを参照してください。
blocking_timeout	トランザクションがロックを獲得するまでの待ち時間 (ミリ秒単位) を返します。「blocking_timeout オプション [データベース]」 431 ページを参照してください。

プロパティ	説明
BytesReceived	クライアント／サーバ通信中に受信したバイト数を返します。
BytesReceivedUncomp	圧縮が無効になっている場合にクライアント／サーバ通信中に受信したバイト数を返します (この値は、圧縮が無効の場合は BytesReceived の値と同じ)。
BytesSent	クライアント／サーバ通信中に送信されたバイト数を返します。
BytesSentUncomp	圧縮が無効になっている場合にクライアント／サーバ通信中に送信されたバイト数を返します (この値は、圧縮が無効の場合は BytesSent の値と同じ)。
CacheHits	成功したキャッシュ読み込み回数を返します。
CacheRead	キャッシュ内で検索されたデータベース・ページの数を返します。
CacheReadIndInt	キャッシュから読み込まれたインデックス内部ノード・ページの数を返します。
CacheReadIndLeaf	キャッシュから読み込まれたインデックス・リーフ・ページの数を返します。
CacheReadTable	キャッシュから読み込まれたテーブル・ページの数を返します。
CarverHeapPages	クエリ最適化などの短期的な目的に使用されたヒープ・ページの数を返します。
chained	BEGIN TRANSACTION 文が指定されていない場合に使用されるトランザクション・モードを返します。「chained オプション [互換性]」 432 ページを参照してください。
CharSet	接続で使用される CHAR 文字セットを返します。
checkpoint_time	データベース・サーバが、あるチェックポイントを実行してから次のチェックポイントを実行するまでの最大時間 (分単位) を返します。「checkpoint_time オプション [データベース]」 432 ページを参照してください。
cis_option	リモート・データ・アクセス用のデバッグ情報がサーバ・メッセージ・ウィンドウに表示される場合は 0 を返し、リモート・データ・アクセス用のデバッグ情報がサーバ・メッセージ・ウィンドウに表示されない場合は 7 を返します。「cis_option オプション [データベース]」 433 ページを参照してください。
cis_rowset_size	各フェッチにおいてリモート・サーバから返される ローの数を返します。「cis_rowset_size オプション [データベース]」 433 ページを参照してください。

プロパティ	説明
ClientLibrary	jConnect 接続の場合は jConnect、Open Client 接続の場合は CT_Library、HTTP 接続の場合は None、ODBC 接続、Embedded SQL 接続、OLE DB 接続、ADO.NET 接続、iAnywhere JDBC ドライバ接続の場合は CmdSeq を返します。
ClientPort	クライアントの TCP/IP ポート番号を返します。接続の種類が TCP/IP でない場合は、0 を返します。
ClientStmtCacheHits	クライアントに文がキャッシュされていることが理由で、この接続には不要な準備文の数を返します。これは、クライアントでの文のキャッシュが無効になった場合に必要となる、追加の準備文の数です。「max_client_statements_cached オプション [データベース]」 463 ページを参照してください。
ClientStmtCacheMisses	この接続において、クライアントのキャッシュに存在した文の中で、再び準備された文の数を返します。これは、キャッシュされた文が再使用できるか検討されたものの、スキーマの変更、データベースのオプション設定の変更、DROP VARIABLE 文によって再使用できなかった回数です。「max_client_statements_cached オプション [データベース]」 463 ページを参照してください。
close_on_endtrans	トランザクションの終了時にカーソルを閉じるかどうかを示す値として、On または Off を返します。「close_on_endtrans オプション [互換性]」 434 ページを参照してください。
collect_statistics_on_dml_updates	INSERT、DELETE、UPDATE などのデータを変更する DML 文の実行中に統計情報を収集するかどうかを示す値として、On または Off を返します。「collect_statistics_on_dml_updates オプション [データベース]」 434 ページを参照してください。
Commit	処理されたコミット要求の数を返します。
CommLink	接続の通信リンクを返します。これは SQL Anywhere がサポートするネットワーク・プロトコルであり、同一コンピュータ接続の場合は local となります。
CommNetworkLink	接続の通信リンクを返します。これは SQL Anywhere がサポートするネットワーク・プロトコルです。値には、SharedMemory、TCPIP、SPX があります。CommNetworkLink プロパティは、同一コンピュータかどうかにかかわらず、常にリンク名を返します。
CommProtocol	Open Client 接続と jConnect 接続の場合は TDS、HTTP 接続の場合は HTTP、ODBC 接続、Embedded SQL 接続、OLE DB 接続、ADO.NET 接続、iAnywhere JDBC ドライバ接続の場合は CmdSeq を返します。

プロパティ	説明
Compression	この接続で通信圧縮が有効であるかどうかを示す値として、On または Off を返します。
conn_auditing	この接続に対して監査が有効になっている場合は、auditing オプションが Off に設定されていても、On を返します。「監査の有効化」 930 ページを参照してください。
connection_authentication	クライアントの認証に使用される文字列を返します。データベースが変更可能になる前に、認証が必要です。「connection_authentication オプション [データベース]」 436 ページを参照してください。
continue_after_raiserror	RAISERROR 文が見つかるたびにプロシージャまたはトリガの実行を停止するかどうかを示す値として、On または Off を返します。「continue_after_raiserror オプション [互換性]」 437 ページを参照してください。
conversion_error	データベースから情報をフェッチするときにデータ型変換障害をレポートするかどうかを示す値として、On または Off を返します。「conversion_error オプション [互換性]」 438 ページを参照してください。
cooperative_commit_timeout	データベース・サーバがディスクへの書き込み前に他の接続の書き込みによって 1 つのログ・ページが満杯になるまで待っている時間 (ミリ秒単位) を返します。「cooperative_commit_timeout オプション [データベース]」 438 ページを参照してください。
cooperative_commits	コミットをいつディスクに書き込むかを示す値として、On または Off を返します。「cooperative_commits オプション [データベース]」 439 ページを参照してください。
CurrentLineNumber	接続で実行されているプロシージャまたは複合文の現在の行番号を返します。実行されているプロシージャの名前は CurrentProcedure プロパティで取得できます。現在の行がクライアント側からの複合文の一部である場合は、空の文字列を返します。
CurrentProcedure	接続で現在実行されているプロシージャの名前を返します。接続でネストされたプロシージャ・コールが実行されている場合は、現在のプロシージャの名前を返します。実行されているプロシージャがない場合は、空の文字列を返します。
Cursor	現在サーバが保持している宣言されたカーソルの数を返します。
CursorOpen	現在サーバが保持している開いたカーソルの数を返します。

プロパティ	説明
database_authentication	データベースの認証に使用される文字列を返します。データベース・サーバの認証が完了してからでなければ、データベースに変更を加えることはできません。「database_authentication [データベース]」 440 ページを参照してください。
date_format	データベースから取り出した日付のフォーマットを示す文字列を返します。「date_format オプション [互換性]」 441 ページを参照してください。
date_order	日付のフォーマット方法を示す文字列を返します。「date_order オプション [互換性]」 443 ページを参照してください。
DBNumber	データベースの ID 番号を返します。
debug_messages	DEBUG ONLY 句を含む MESSAGE 文を実行するかどうかを示す値として、On または Off を返します。「debug_messages オプション [データベース]」 443 ページを参照してください。
dedicated_task	要求処理タスクがこの接続からの要求の処理専用であるかどうかを示す値として、On または Off を返します。「dedicated_task オプション [データベース]」 444 ページを参照してください。
default_dbspace	デフォルト DB 領域の名前を返します。デフォルト DB 領域が指定されていない場合は、空の文字列を返します。「default_dbspace オプション [データベース]」 444 ページを参照してください。
default_timestamp_increment	TIMESTAMP 型のカラム内の値を一意に保つために加算する値 (ミリ秒単位) を返します。「default_timestamp_increment オプション [データベース] [Mobile Link クライアント]」 445 ページを参照してください。
delayed_commit_timeout	COMMIT の後でデータベース・サーバがアプリケーションに制御を戻すまでの待ち時間 (ミリ秒単位) を返します。「delayed_commit_timeout オプション [データベース]」 446 ページを参照してください。
delayed_commits	COMMIT の後でデータベース・サーバがアプリケーションに制御を戻す時点を示す値として、On または Off を返します。「delayed_commits オプション [データベース]」 446 ページを参照してください。
DiskRead	ディスクから読み込まれたページの数を返します。
DiskReadIndInt	ディスクから読み込まれたインデックス内部ノード・ページの数を返します。

プロパティ	説明
DiskReadIndLeaf	ディスクから読み込まれたインデックス・リーフ・ページの数を返します。
DiskReadTable	ディスクから読み込まれたテーブル・ページの数を返します。
DiskWrite	ディスクに書き込まれた修正ページの数を返します。
divide_by_zero_error	ゼロ除算によってエラーが発生する場合は On を返し、ゼロ除算によってエラーが発生しない場合は Off を返します。「divide_by_zero_error オプション [互換性]」 448 ページを参照してください。
Encryption	接続が暗号化されるかどうかを示す値を返します。「Encryption 接続パラメータ [ENC]」 247 ページを参照してください。
escape_character	このプロパティはシステムで使用するために予約されています。このオプションの設定は変更しないでください。
EventName	接続でイベント・ハンドラが実行されている場合、関連付けられているイベントの名前を返します。それ以外の場合、結果は NULL です。
exclude_operators	このプロパティはシステムで使用するために予約されています。このオプションの設定は変更しないでください。
ExprCacheAbandons	ヒット率が低すぎるために式キャッシュが中止された回数を返します。
ExprCacheDropsToReadOnly	ヒット率が低いために式キャッシュが読み込み専用ステータスに変更された回数を返します。
ExprCacheEvicts	式キャッシュからの出力数を返します。
ExprCacheHits	式キャッシュ内のヒット数を返します。
ExprCacheInserts	式キャッシュに挿入された値の数を返します。
ExprCacheLookups	式キャッシュ内で実行されたルックアップの回数を返します。
ExprCacheResumesOfReadWrite	ヒット率の上昇によって式キャッシュが読み込み／書き込みステータスに戻った回数を返します。
ExprCacheStarts	式キャッシュが開始された回数を返します。
extended_join_syntax	マルチテーブル・ジョインについて重複する相関名構文のクエリを許可するか、またはエラーとしてレポートするかを示す値を返します。「extended_join_syntax オプション [データベース]」 449 ページを参照してください。

プロパティ	説明
fire_triggers	データベースにおいてトリガが起動する場合は On を返し、起動しない場合は Off を返します。「fire_triggers オプション [互換性]」 450 ページを参照してください。
first_day_of_week	週の最初の曜日を表す番号を返します。7 は日曜日を表し、1 は月曜日を表します。「first_day_of_week オプション [データベース]」 450 ページを参照してください。
float_as_double	SQL 文においてキーワード FLOAT のオカレンスをキーワード DOUBLE と同義と解釈する場合は On を返し、それ以外の場合は Off を返します。「float_as_double オプション [互換性]」 451 ページを参照してください。
for_xml_null_treatment	NULL 値を含む要素と属性が結果から除外される場合は OMIT を返し、クエリ内で FOR XML 句が使用されている場合に NULL 値に対応する空の要素または属性が生成される場合は EMPTY を返します。「for_xml_null_treatment オプション [データベース]」 452 ページを参照してください。
force_view_creation	このプロパティはシステムで使用するために予約されています。このオプションの設定は変更しないでください。
FullCompare	インデックスのハッシュ値を超えて実行された比較の回数を返します。
GetData	GETDATA 要求の数を返します。
global_database_id	DEFAULT GLOBAL AUTOINCREMENT が指定されたカラムの開始値を返します。「global_database_id オプション [データベース]」 453 ページを参照してください。
HashForcedPartitions	メモリ競合が原因でハッシュ演算子が分割された回数を返します。
HashRowsFiltered	プローブ・ローがビットベクトル・フィルタによって拒否される割合を返します。
HashRowsPartitioned	ローがハッシュ・ワーク・テーブルに書き込まれる割合を返します。
HashWorkTables	ハッシュベースの演算用に作成されたワーク・テーブルの数を返します。
HeapsCarver	クエリ最適化などの短期的な目的に使用されたヒープの数を返します。
HeapsLocked	キャッシュ内で現在ロックされている再配置可能なヒープの数を返します。
HeapsQuery	クエリ処理 (ハッシュ操作とソート操作) に使用されるヒープの数を返します。

プロパティ	説明
HeapsRelocatable	再配置可能なヒープの数を返します。
http_session_timeout	現在の HTTP セッション・タイムアウトを返します (分単位)。「http_session_timeout オプション [データベース]」 453 ページを参照してください。
HttpServiceName	Web アプリケーションのサービス名オリジンを返します。このプロパティは、エラー・レポートやフロー制御の場合に便利です。このプロパティを HTTP 要求以外によって実行されたストアド・プロシージャから選択した場合、または接続が現在アクティブではなく、HTTP セッションの再開を待っている状態である場合は、空の文字列を返します。
IdleTimeout	接続のアイドル・タイムアウト値を返します。「Idle 接続パラメータ」 251 ページを参照してください。
IndAdd	インデックスに追加されたエントリの数を返します。
IndLookup	インデックス内で検索されたエントリの数を返します。
integrated_server_name	統合化ログインを目的とする Windows ユーザ・グループ・メンバシップの検索に使用される Domain Controller サーバの名前を返します。「integrated_server_name オプション [データベース]」 454 ページを参照してください。
isolation_level	接続の独立性レベルを返します。「isolation_level オプション [互換性]」 455 ページを参照してください。
java_location	データベースに対して Java VM が指定されている場合、そのパスを返します。
java_main_userid	接続をクラスのインストールや他の Java 関連の管理タスクに使用できるデータベース・ユーザの名前を返します。
Language	ロケール言語を返します。
LastIdle	要求間のチック数を返します。
LastPlanText	接続で最後に実行されたクエリの長いテキスト・プランを返します。最後のプランを記憶するかどうかを指定するには、sa_server_option システム・プロシージャの RememberLastPlan オプションまたは -zp サーバ・オプションを使用します。「-zp サーバ・オプション」 213 ページを参照してください。
LastReqTime	指定された接続において最後の要求が開始された時刻を返します。

プロパティ	説明
LastStatement	現在の接続で最後に準備された SQL 文を返します。「-zl サーバ・オプション」 211 ページを参照してください。 LastStatement の値は、文が準備されると同時に設定され、文が削除されると同時にクリアされます。各接続につき 1 つの文の文字列のみが記憶されます。 ある接続について sa_conn_activity が空でない値を返した場合、その接続で現在実行されている文である可能性が高くなります。その文が完了している場合は、文がすでに削除され、このプロパティの値がクリアされている可能性があります。アプリケーションが複数の文を準備し、それらの文のステートメント・ハンドルを保持している場合、LastStatement が返す値は接続で現在実行されている処理を表しません。 クライアントでの文のキャッシュが有効であり、キャッシュされた文が再使用されているとき、このプロパティは空の文字列を返します。
LivenessTimeout	現在の接続の活性タイムアウト時間を返します。「LivenessTimeout 接続パラメータ [LTO]」 255 ページを参照してください。
lock_rejected_rows	このプロパティはシステムで使用するために予約されています。このオプションの設定は変更しないでください。
LockCount	接続で保持されているロックの数を返します。
LockName	接続が解放を待っているロックを示す 64 ビット符号なし整数を返します。
LockTableOID	接続がブロックされていない場合、または接続が CONNECTION_PROPERTY を呼び出した接続とは別のデータベース上にある場合は、0 を返します。それ以外の場合は、接続が解放を待っているロックが設定されているテーブルのオブジェクト ID を返します。このオブジェクト ID によって、SYSTAB システム・ビューを使用してテーブル情報を検索できます。「SYSTAB システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
log_deadlocks	デッドロック情報がレポートされる場合は On を返し、それ以外の場合は Off を返します。「log_deadlocks オプション [データベース]」 458 ページを参照してください。
LogFreeCommit	Redo Free Commit の数を返します。Redo Free Commit が発生するのは、トランザクション・ログのコミットが要求されているが、ログはすでに書き込まれている (コミットが実行されている) 場合です。

プロパティ	説明
login_mode	統合化ログインと Kerberos がサポートされているかどうかを示す値として、Standard、Integrated、Kerberos のうち、いずれか 1 つまたは複数を返します。「login_mode オプション [データベース]」 459 ページを参照してください。
login_procedure	起動時に互換性オプションを設定するストアド・プロシージャの名前を返します。「login_procedure オプション [データベース]」 460 ページを参照してください。
LoginTime	接続が確立された日付と時刻を返します。
LogWrite	トランザクション・ログに書き込まれたページの数を返します。
materialized_view_optimization	実体化ビュー (Materialized View) をクエリ最適化の実行中に使用するかどうかを示す値を返します。「materialized_view_optimization オプション [データベース]」 462 ページを参照してください。
max_client_statements_cached	クライアントでキャッシュされる文の数を返します。「max_client_statements_cached オプション [データベース]」 463 ページを参照してください。
max_cursor_count	接続で一度に使用できるカーソルの最大数を返します。「max_cursor_count オプション [データベース]」 464 ページを参照してください。
max_plans_cached	キャッシュに格納される実行プランの最大数を返します。「max_plans_cached オプション [データベース]」 465 ページを参照してください。
max_query_tasks	データベース・サーバがクエリの処理に使用できる要求の最大数を返します。「max_query_tasks オプション [データベース]」 466 ページを参照してください。
max_recursive_iterations	再帰共通テーブル式が実行できる反復処理の最大回数を返します。「max_recursive_iterations オプション [データベース]」 467 ページを参照してください。
max_statement_count	接続で一度に使用できる準備文の最大数を返します。「max_statement_count オプション [データベース]」 467 ページを参照してください。
max_temp_space	1 つの接続で使用できるテンポラリ・ファイル領域の最大サイズを返します。「max_temp_space オプション [データベース]」 469 ページを参照してください。
MessageReceived	MESSAGE 文によって生成され、WAITFOR 文が中断する原因となった文字列を返します。それ以外の場合は、空の文字列が返されます。

プロパティ	説明
min_password_length	データベースにおける新しいパスワードの最小長を返します。「min_password_length オプション [データベース]」 470 ページを参照してください。
Name	現在の接続の名前を返します。
NcharCharSet	接続で使用される NCHAR 文字セットを返します。
nearest_century	文字列から日付への変換において 2 桁の年がどのように解釈されるかを示す値を返します。「nearest_century オプション [互換性]」 470 ページを参照してください。
NodeAddress	クライアント／サーバ接続のクライアント側に対応するノードを返します。クライアントとサーバの両方が同じコンピュータにある場合は、空の文字列を返します。
non_keywords	オフになっているため識別子として使用できるキーワードのリストを返す (そのようなキーワードが存在する場合)。「non_keywords オプション [互換性]」 471 ページを参照してください。
Number	接続の ID 番号を返します。
odbc_describe_binary_as_varbinary	SQL Anywhere ODBC ドライバが BINARY カラムと VARBINARY カラムをどちらも SQL_BINARY として記述する場合は Off を返し、ODBC ドライバが BINARY カラムと VARBINARY カラムを SQL_VARBINARY として記述する場合は On を返します。「odbc_describe_binary_as_varbinary [データベース]」 471 ページを参照してください。
odbc_distinguish_char_and_varchar	CHAR カラムが SQL_VARCHAR として記述される場合は Off を返し、SQL_CHAR として記述される場合は On を返します。「odbc_distinguish_char_and_varchar オプション [データベース]」 472 ページを参照してください。
oem_string	データベース・ファイルのヘッダ・ページ内の文字列を返します。「oem_string オプション [データベース]」 473 ページを参照してください。
on_charset_conversion_failure	文字セットの変換中にエラーが発生した場合の動作を示す値として、Ignore、Warning、Error のいずれかを返します。「on_charset_conversion_failure オプション [データベース]」 475 ページを参照してください。
on_tsql_error	ストアド・プロシージャの実行中にエラーが発生した場合の動作を示す値として、Stop、ConditionalAL、Continue のいずれかを返します。「on_tsql_error オプション [互換性]」 475 ページを参照してください。

プロパティ	説明
optimistic_wait_for_commit	wait_for_commit オプションのロック動作を示す値として、On または Off を返します。 「optimistic_wait_for_commit オプション [互換性]」 476 ページを参照してください。
optimization_goal	クエリ処理を最適化する方法を示す値として、First-row または All-rows を返します。「optimization_goal オプション [データベース]」 477 ページを参照してください。
optimization_level	0 ー 15 の範囲の値を返します。この値を使用して、SQL Anywhere クエリ・オプティマイザが SQL 文のアクセス・プランの検索に費やす作業量を指定します。 「optimization_level オプション [データベース]」 478 ページを参照してください。
optimization_workload	SQL Anywhere クエリ・オプティマイザが SQL 文のアクセス・プランの検索に費やす作業量を示す値を返します。 「optimization_workload オプション [データベース]」 479 ページを参照してください。
PacketSize	接続で使用されるパケット・サイズ (バイト単位) を返します。
PacketsReceived	受信したクライアント／サーバ通信パケットの数を返します。
PacketsReceivedUncomp	圧縮が無効になっている場合にクライアント／サーバ通信中に受信したパケットの数を返す (この値は、圧縮が無効の場合は PacketsReceived の値と同じ)。
PacketsSent	送信されたクライアント／サーバ通信パケットの数を返します。
PacketsSentUncomp	圧縮が無効になっている場合にクライアント／サーバ通信中に送信されたパケットの数を返す (この値は、圧縮が無効の場合は PacketsSent の値と同じ)。
percent_as_comment	コメント・マーカとしてパーセント記号 (%) が使用される場合は On を返し、それ以外の場合は Off を返します。 「percent_as_comment オプション [互換性]」 480 ページを参照してください。
pinned_cursor_percent_of_cache	カーソルを固定するために使用できるキャッシュの割合を返します。「pinned_cursor_percent_of_cache オプション [データベース]」 481 ページを参照してください。
post_login_procedure	ユーザが接続したときにアプリケーションが表示する必要のあるメッセージがプロシージャの結果セットに含まれている場合、そのプロシージャの名前を返します。 「post_login_procedure [データベース]」 481 ページを参照してください。

プロパティ	説明
precision	10 進数精度と数値精度の設定を返します。「precision オプション [データベース]」 483 ページを参照してください。
prefetch	プリフェッチが行われない場合は Off、カーソル・タイプが SENSITIVE であるかクエリにプロキシ・テーブルが含まれていないかぎりプリフェッチが行われる場合は Conditional、カーソル・タイプが SENSITIVE であり、カーソルにプロキシ・テーブルが含まれていても常にプリフェッチが行われる場合は Always を返します。「prefetch オプション [データベース]」 483 ページを参照してください。
Prepares	実行された文の準備作業の数を返します。
PrepStmt	現在サーバが保持している準備文の数を返します。
preserve_source_format	プロシージャ、トリガ、ビュー、イベント・ハンドラの元のソース定義がシステム・ファイルに保存される場合は On を返し、それ以外の場合は Off を返します。「preserve_source_format オプション [データベース]」 484 ページを参照してください。
prevent_article_pkey_update	パブリケーションで使用されるテーブルのプライマリ・キー・カラムの更新が許可されている場合は On を返し、それ以外の場合は Off を返します。「prevent_article_pkey_update オプション [データベース] [Mobile Link クライアント]」 485 ページを参照してください。
query_plan_on_open	カーソルを開くときにプランが返されるかどうかを示す値を返します。「query_plan_on_open オプション [互換性]」 486 ページを参照してください。
QueryBypassed	オプティマイザ・バイパスによって最適化された要求の数を返します。
QueryCachedPlans	接続において現在キャッシュされているクエリ実行プランの数を返します。
QueryCachePages	実行プランの保存に使用されるキャッシュ・ページの数を返します。
QueryHeapPages	クエリ処理 (ハッシュ操作とソート操作) に使用されるキャッシュ・ページの数を返します。
Query JHToJNLOptUsed	ハッシュ・ジョインがネスト・ループ・ジョインに変換された回数を返します。
QueryLowMemoryStrategy	メモリ不足が原因でサーバが実行プランを実行中に変更した回数を返します。使用できるメモリがオプティマイザの推定よりも少ない、または実行プランが必要とするメモリがオプティマイザの推定よりも多い場合に、プランが変更されることがあります。

プロパティ	説明
QueryOptimized	完全に最適化された要求の数を返します。
QueryReused	プラン・キャッシュから再利用された要求の数を返します。
QueryRowsBufferFetch	バッファを使用してフェッチされたローの数を返します。
QueryRowsMaterialized	クエリ処理中にローがワーク・テーブルに書き込まれる割合を返します。
quoted_identifier	二重引用符で囲まれた文字列が識別子として解釈される場合は On を返し、リテラル文字列として解釈される場合は Off を返します。「quoted_identifier オプション [互換性]」 487 ページを参照してください。
read_past_deleted	独立性レベル 1 または 2 の逐次スキャンにおいてコミットされていない削除ローが無視される場合は On を返し、独立性レベル 1 または 2 の逐次スキャンがコミットされていない削除ローでブロックされる場合は Off を返します。「read_past_deleted オプション [データベース]」 487 ページを参照してください。
recovery_time	データベース・サーバがシステム障害から回復するのに要する最長時間 (分単位) を返します。「recovery_time オプション [データベース]」 488 ページを参照してください。
RecursiveIterations	再帰ユニオンの反復回数を返します。
RecursiveIterationsHash	再帰ハッシュ・ジョインでハッシュ方式が使用された回数を返します。
RecursiveIterationsNested	再帰ハッシュ・ジョインでネスト・ループ方式が使用された回数を返します。
RecursiveJNLMisses	再帰ハッシュ・ジョインでのインデックス・プローブ・キャッシュ・ミスの数を返します。
RecursiveJNLProbes	再帰ハッシュ・ジョインにおけるインデックス・プローブの試行回数を返します。
remote_idle_timeout	Web サービスのクライアント・プロシージャとファンクションで許容される休止時間 (秒単位) を返します。「remote_idle_timeout オプション [データベース]」 488 ページを参照してください。
replicate_all	データベースが Replication Server インストール環境のプライマリ・サイトとして動作する場合は On を返し、それ以外の場合は Off を返します。「replicate_all オプション [Replication Agent]」 489 ページを参照してください。

プロパティ	説明
ReqCountActive	処理が完了した要求の数を返します。RequestTiming サーバ・プロパティが Off に設定されている場合は、NULL を返します。「-zt オプション」 216 ページを参照してください。
ReqCountBlockContention	接続がアトミック・アクセスを待った回数を返します。-zt オプションが指定されていない場合は、NULL を返します。「-zt オプション」 216 ページを参照してください。
ReqCountBlockIO	接続が I/O 処理の完了を待った回数を返します。-zt オプションが指定されていない場合は、NULL を返します。「-zt オプション」 216 ページを参照してください。
ReqCountBlockLock	接続がロックの解放を待った回数を返します。-zt オプションが指定されていない場合は、NULL を返します。「-zt オプション」 216 ページを参照してください。
ReqCountUnscheduled	接続がスケジューリングを待った回数を返します。-zt オプションが指定されていない場合は、NULL を返します。「-zt オプション」 216 ページを参照してください。
ReqStatus	要求のステータスを返します。値は次のいずれかです。 ◆ **Idle**　この接続では現在、要求を処理していない。 ◆ **Unscheduled***　この接続では他の処理が実行されており、ワーカ・スレッドの解放を待っている。 ◆ **BlockedIO***　この接続はブロックされ、I/O 処理の完了を待っている。 ◆ **BlockedContention***　この接続はブロックされ、共有データベース・サーバ・データ構造体へのアクセスを待っている。 ◆ **BlockedLock**　この接続はブロックされ、オブジェクトのロックの解放を待っている。 ◆ **Executing**　この接続では現在、要求を実行している。 アスタリスク (*) が付いている値が返されるのは、-zt サーバ・オプションによってデータベース・サーバで要求タイミング情報のロギングがオンになっている場合に限られます。要求タイミング情報のロギングが実行されていない場合は (デフォルト)、Executing が返されます。 詳細については、「-zt オプション」 216 ページを参照してください。
ReqTimeActive	要求の処理に要した時間を返します。-zt オプションが指定されていない場合は、NULL を返します。「-zt オプション」 216 ページを参照してください。

プロパティ	説明
ReqTimeBlockContention	アトミック・アクセスを取得するまでの待ち時間を返します。RequestTiming サーバ・プロパティが Off に設定されている場合は、NULL を返します。「-zt オプション」 216 ページを参照してください。
ReqTimeBlockIO	I/O 処理が完了するまでの待ち時間を返します。-zt オプションが指定されていない場合は、NULL を返します。「-zt オプション」 216 ページを参照してください。
ReqTimeBlockLock	ロックが解放されるまでの待ち時間を返します。-zt オプションが指定されていない場合は、NULL を返します。「-zt オプション」 216 ページを参照してください。
ReqTimeUnscheduled	未スケジュール時間を返します。-zt オプションが指定されていない場合は、NULL を返します。「-zt オプション」 216 ページを参照してください。
ReqType	最後の要求のタイプを返します。
RequestsReceived	クライアント／サーバ通信要求またはラウンド・トリップの数を返します。このプロパティでは、PacketsReceived とは異なり、マルチパケット要求を 1 つの要求として数え、活性パケットを計数の対象から除外します。
return_date_time_as_string	日付、時刻、タイムスタンプの値が文字列としてアプリケーションに返される場合は On を返し、日付または時刻データ型として返される場合は Off を返します。「return_date_time_as_string オプション [データベース]」 491 ページを参照してください。
ri_trigger_time	参照整合性アクションを UPDATE または DELETE の前後いずれの時点で実行するかを示す値として、Before または After を返します。「ri_trigger_time オプション [互換性]」 492 ページを参照してください。
Rlbk	処理されたロールバック要求の数を返します。
rollback_on_deadlock	参照整合性アクションが UPDATE または DELETE の後に実行される場合は After を返し、UPDATE または DELETE の前に実行される場合は Before を返します。「rollback_on_deadlock [データベース]」 492 ページを参照してください。
RollbackLogPages	ロールバック・ログのページ数を返します。
row_counts	ローの数が常に正確である場合は On を返し、ローの数が通常は推定値である場合は Off を返します。「row_counts オプション [データベース]」 493 ページを参照してください。
scale	接続における 10 進数と数値の位取りを返します。「scale オプション [データベース]」 494 ページを参照してください。

プロパティ	説明
secure_feature_key	データベース・サーバにおいて機能を有効または無効にするキーを格納します。このプロパティの値を選択すると、常に空の文字列が返されます。
ServerPort	データベース・サーバの TCP/IP ポート番号または 0 を返します。
SessionCreateTime	HTTP セッションが作成された時刻を返します。
SessionID	接続のセッション ID が定義されている場合は、そのセッション ID を返し、それ以外の場合は空の文字列を返します。
SessionLastTime	HTTP セッションにおける最後の要求の時刻を返します。
SnapshotCount	接続に関連付けられているスナップショットの数を返します。
sort_collation	ORDER BY 句が変更されていない場合は Internal を返し、それ以外の場合は照合名または照合 ID を返します。「sort_collation オプション [データベース]」 495 ページを参照してください。
SortMergePasses	ソート中に使用されたマージ・パスの数を返します。
SortRowsMaterialized	ローがソート・ワーク・テーブルに書き込まれる割合を返します。
SortRunsWritten	ソート中に書き込まれたソート実行の数を返します。
SortSortedRuns	実行の生成中に作成されたソート実行の数を返します。
SortWorkTables	ソート用に作成されたワーク・テーブルの数を返します。
sql_flagger_error_level	指定された SQL/2003 に含まれていない場合にエラーとして通知する SQL を示す値として、次のいずれかを返します。 ◆ **E** 初級レベル SQL/2003 構文でない構文を通知します。 ◆ **I** 中級レベル SQL/2003 構文でない構文を通知します。 ◆ **F** 上級 SQL/2003 構文でない構文を通知します。 ◆ **W** サポートされている構文をすべて許可します。 詳細については、「sql_flagger_error_level オプション [互換性]」 496 ページを参照してください。

プロパティ	説明
sql_flagger_warning_level	指定された SQL/2003 に含まれていない場合に警告として通知する SQL を示す値として、次のいずれかを返します。 ♦ **E** 初級レベル SQL/2003 構文でない構文を通知します。 ♦ **I** 中級レベル SQL/2003 構文でない構文を通知します。 ♦ **F** 上級 SQL/2003 構文でない構文を通知します。 ♦ **W** サポートされている構文をすべて許可します。 詳細については、「sql_flagger_warning_level オプション [互換性]」 497 ページを参照してください。
string_rtruncation	文字列がトランケートされたときにエラーが発生する場合は On を返し、エラーなしで文字列がトランケートされる場合は Off を返します。「string_rtruncation オプション [互換性]」 500 ページを参照してください。
subsume_row_locks	データベース・サーバがテーブルに対する個別ロー・ロックを取得する場合は On を返し、それ以外の場合は Off を返します。「subsume_row_locks オプション [データベース]」 501 ページを参照してください。
suppress_tds_debugging	TDS デバッグ情報がサーバ・メッセージ・ウィンドウに表示される場合は Off を返し、表示されない場合は On を返します。「suppress_tds_debugging オプション [データベース]」 502 ページを参照してください。
synchronize_mirror_on_commit	データベース・ミラー・サーバがコミット時に同期される場合は On を返し、それ以外の場合は Off を返します。「synchronize_mirror_on_commit オプション [データベース]」 502 ページを参照してください。
tds_empty_string_is_null	TDS 接続において空の文字列が NULL として返される場合は On を返し、TDS 接続において 1 つのブランク文字を含む文字列が返される場合は Off を返します。「tds_empty_string_is_null オプション [データベース]」 503 ページを参照してください。
temp_space_limit_check	データベース・サーバが接続で使用できるテンポラリ領域のサイズをチェックする場合は On を返し、データベース・サーバが接続で使用できる領域のサイズをチェックしない場合は Off を返します。「temp_space_limit_check オプション [データベース]」 503 ページを参照してください。
TempTablePages	テンポラリ・テーブルで使用されるテンポラリ・ファイルのページ数を返します。

プロパティ	説明
time_format	データベースから取り出される時刻の文字列フォーマットを返します。「time_format オプション [互換性]」 504 ページを参照してください。
timestamp_format	接続のローカル時間を表示するために協定世界時 (UTC: Coordinated Universal Time) に加算する必要がある時間 (分単位) を返します。「timestamp_format オプション [互換性]」 505 ページを参照してください。
TimeZoneAdjustment	接続のローカル時間を表示するために協定世界時 (UTC: Coordinated Universal Time) に加算する必要がある時間 (分単位) を返します。「time_zone_adjustment オプション [データベース]」 505 ページを参照してください。
TransactionStartTime	COMMIT または ROLLBACK の後にデータベースが最初に変更された時刻を含む文字列を返します。最後の COMMIT または ROLLBAK 以降にデータベースが変更されていない場合は、空の文字列を返します。
truncate_timestamp_values	タイムスタンプ値で使用される小数点以下の桁数に制限がある場合は On を返し、それ以外の場合は Off を返します。「truncate_timestamp_values オプション [データベース] [Mobile Link クライアント]」 507 ページを参照してください。
truncate_with_auto_commit	COMMIT が TRUNCATE TABLE 文の実行の前後に実行される場合は On を返し、それ以外の場合は Off を返します。「truncate_with_auto_commit オプション [データベース]」 508 ページを参照してください。
tsql_hex_constant	16 進定数がバイナリ文字定数として処理される場合は On を返し、それ以外の場合は Off を返します。「tsql_hex_constant オプション [互換性]」 509 ページを参照してください。
tsql_outer_joins	DML 文で Transact-SQL 外部ジョインが使用できるかどうかを示す値を返します。「tsql_outer_joins オプション [互換性]」 509 ページを参照してください。
tsql_variables	Embedded SQL のホスト変数名のプレフィクスとしてコロンの代わりに @ 符号を使用できる場合は On を返し、それ以外の場合は Off を返します。「tsql_variables オプション [互換性]」 509 ページを参照してください。
UncommitOp	コミットされていない操作の数を返します。
update_statistics	このプロパティはシステムで使用するために予約されています。このオプションの設定は変更しないでください。
upgrade_database_capability	このプロパティはシステムで使用するために予約されています。このオプションの設定は変更しないでください。

プロパティ	説明
user_estimates	クエリ・オプティマイザがクエリの述部に含まれる選択性推定を尊重するか無視するかを示す値として、次のいずれかを返します。 ♦ **Enabled** ユーザが提供する選択性推定をすべて尊重します。このオプションは、On を使用して有効にすることもできます。 ♦ **Override-Magic** ユーザ選択性推定は尊重されますが、オプティマイザが最後の手段であるヒューリスティック値 (マジック値とも呼ばれます) の使用を選択する以外に方法がない場合のみ使用されます。 ♦ **Disabled** 他の推定データが使用できないときは、ユーザ推定は無視され、マジック値が使用されます。このオプションは、Off を使用して無効にすることもできます。 詳細については、「user_estimates オプション [データベース]」 511 ページを参照してください。
UserAppInfo	AppInfo 接続パラメータによって接続文字列に指定された文字列を返します。 詳細については、「AppInfo 接続パラメータ [APP]」 231 ページを参照してください。
UserID	接続のユーザ ID を返します。
UtilCmdsPermitted	この接続で CREATE DATABASE、DROP DATABASE、RESTORE DATABASE などのユーティリティ・コマンドの使用が許可されているかどうかを示す値として、On または Off を返します。「-gu サーバ・オプション」 176 ページを参照してください。
uuid_has_hyphens	UUID に 4 つのハイフンが含まれている場合は On を返し、ハイフンが含まれていない場合は Off を返します。「uuid_has_hyphens オプション [データベース]」 512 ページを参照してください。
verify_password_function	パスワードの検証に使用される関数が指定されている場合は、その関数名を返します。「verify_password_function オプション [データベース]」 513 ページを参照してください。
wait_for_commit	データベースが次の COMMIT 文まで外部キー整合性をチェックしない場合は、On を返します。それ以外の場合は Off を返します。この場合、check_on_commit オプションで作成されていないすべての外部キーが、挿入、更新、削除時にチェックされます。「wait_for_commit オプション [データベース]」 515 ページを参照してください。

プロパティ	説明
webservice_namespace_host	生成された WSDL ドキュメント内で XML ネームスペースとして使用されるホスト名を返す (そのようなホスト名が指定されている場合)。「webservice_namespace_host オプション [データベース]」 515 ページを参照してください。

サーバ・レベルのプロパティ

次の表に、サーバ全体に適用できるプロパティを示します。

例

- **サーバ・プロパティの値を取り出すには、次の手順に従います。**

  · property システム関数を使用します。たとえば、次の文はグローバル・サーバ・データ構造に使用されたキャッシュ・ページ数を返します。

  ```
  SELECT PROPERTY ( 'MainHeapPages' );
  ```

- **すべてのサーバ・プロパティの値を取り出すには、次の手順に従います。**

  · sa_eng_properties システム・プロシージャを使用します。

  ```
  CALL sa_eng_properties;
  ```

参照

- 「接続レベルのプロパティ」 518 ページ
- 「データベース・レベルのプロパティ」 550 ページ

説明

プロパティ	説明
ActiveReq	現在要求を処理中のサーバ・スレッドの数を返します。
AvailIO	現在使用可能な I/O 制御ブロックの数を返します。
BuildChange	予約。
BuildClient	予約。
BuildProduction	データベース・サーバが、実際の運用で使用するためにコンパイルされている場合は Yes を返し、デバッグ用のビルドである場合は No を返します。
BuildReproducible	予約。
BytesReceived	クライアント／サーバ通信中に受信したバイト数を返します。

プロパティ	説明
BytesReceivedUncomp	圧縮が無効になっている場合にクライアント／サーバ通信中に受信したバイト数を返します (この値は、圧縮が無効の場合は BytesReceived の値と同じ)。
BytesSent	クライアント／サーバ通信中に送信されたバイト数を返します。
BytesSentUncomp	圧縮が無効になっている場合にクライアント／サーバ通信中に送信されたバイト数を返します (この値は、圧縮が無効の場合は BytesSent の値と同じ)。
CacheAllocated	サーバ・データ構造に割り付けられたキャッシュ・ページの数を返します。
CacheFile	データベース・ファイルから取得したデータを格納するキャッシュ・ページの数を返します。
CacheFileDirty	ダーティな (書き込みが必要な) キャッシュ・ページの数を返します。
CacheFree	未使用のキャッシュ・ページ数を返します。
CacheHitsEng	データベース・ページのルックアップ回数を返します。
CachePanics	キャッシュ・マネージャが割り付けるページの検索に失敗した回数を返します。
CachePinned	固定キャッシュ・ページ数を返します。
CacheReadEng	キャッシュの読み込み数を返します。
CacheReplacements	キャッシュ内の置換されたページの数を返します。
CacheScavenges	キャッシュ・マネージャが割り付けるページをスカベンジした回数を返します。
CacheScavengeVisited	割り付けるページのスカベンジ中にアクセスしたページの数を返します。
CacheSizingStatistics	キャッシュのサイズを変更するときにサーバがキャッシュ・サイズ設定の統計を表示する場合は Yes を返し、それ以外の場合は No を返します。「-cs サーバ・オプション」 156 ページを参照してください。
CarverHeapPages	クエリ最適化などの短期的な目的に使用されたヒープ・ページの数を返します。
CharSet	データベース・サーバが使用している CHAR 文字セットを返します。

プロパティ	説明
ClientStmtCacheHits	クライアントの文がキャッシュされたために不要となった準備文の数を返します。これは、クライアントでの文のキャッシュが無効になった場合に必要となる、追加の準備文の数です。「max_client_statements_cached オプション [データベース]」 463 ページを参照してください。
ClientStmtCacheMisses	クライアントのキャッシュに存在した文の中で、再び準備された文の数を返します。これは、キャッシュされた文が再使用できるか検討されたものの、スキーマの変更、データベースのオプション設定の変更、DROP VARIABLE 文によって再使用できなかった回数です。「max_client_statements_cached オプション [データベース]」 463 ページを参照してください。
CollectStatistics	データベース・サーバがパフォーマンス統計を収集するかどうかを示す値として、Yes または No を返します。「-k サーバ・オプション」 177 ページを参照してください。
CommandLine	データベース・サーバの起動に使用されたコマンド・ラインを返します。 -ek オプションを使用してデータベース用の暗号化キーを指定した場合、キーはこのプロパティで返される値にあるアスタリスクの定数文字列で置き換えられます。
CompactPlatformVer	PlatformVer プロパティの縮小バージョンを返します。
CompanyName	このソフトウェアを所有している会社の名前を返します。
ConnsDisabled	新しい接続を禁止するサーバ・オプションの現在の設定を示す値として、Yes または No を返します。「sa_server_option システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
ConsoleLogFile	-o オプションが指定されている場合は、サーバ・メッセージ・ウィンドウからのメッセージがロギングされるファイルの名前を返します。それ以外の場合は、空の文字列を返します。「-o サーバ・オプション」 182 ページを参照してください。
ConsoleLogMaxSize	サーバ・メッセージ・ウィンドウ情報のロギングに使用するファイルの最大サイズ (バイト単位) を返します。「-os サーバ・オプション」 184 ページを参照してください。
CurrentCacheSize	現在のキャッシュ・サイズ (キロバイト単位) を返します。
DebuggingInformation	サーバがトラブルシューティング用の診断メッセージを表示する場合は Yes を返し、それ以外の場合は No を返します。「-z サーバ・オプション」 210 ページを参照してください。
DefaultCollation	新規データベースに使用される照合を返します (明示的に指定されている照合がない場合)。

プロパティ	説明
DefaultNcharCollation	サーバ・コンピュータのデフォルト NCHAR 照合の名前を返します (ICU がインストールされている場合は UCA、インストールされていない場合は UTF8BIN)。
DiskReadEng	ディスクの読み込み数を返します。
ExchangeTasks	クエリの並列実行で現在使用されているタスクの数を返します。
ExchangeTasksCompleted	データベース・サーバが起動された後、クエリ内並列処理に使用された内部タスクの総数を返します。「クエリ実行時の並列処理」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
FipsMode	データベース・サーバの起動時に -fips オプションが使用された場合は Yes を返し、それ以外の場合は No を返します。
FirstOption	データベース・オプションに対応する最初の接続プロパティを表す番号を返します。
FreeBuffers	使用できるネットワーク・バッファの数を返します。
FunctionMaxParms	関数で指定可能なパラメータの最大数を返します。関数は、正の整数である *function-number* によって指定される値で識別されます。次に例を示します。 SELECT PROPERTY ('FunctionMaxParms', *function-number*); *function-number* は、リリース間で変更されることがあることに注意してください。
FunctionMinParms	関数で指定しなければならないパラメータの最小数を返します。関数は、正の整数である *function-number* によって指定される値で識別されます。次に例を示します。 SELECT PROPERTY ('FunctionMaxParms', *function-number*); *function-number* は、リリース間で変更されることがあることに注意してください。
FunctionName	正の整数である *function-number* によって指定される値で識別される関数名を返します。 SELECT PROPERTY ('FunctionName', *function-number*); *function-number* は、リリース間で変更されることがあることに注意してください。
HeapsCarver	クエリ最適化などの短期的な目的に使用されたヒープの数を返します。
HeapsLocked	キャッシュ内で現在ロックされている再配置可能なヒープの数を返します。

プロパティ	説明
HeapsQuery	クエリ処理 (ハッシュ操作とソート操作) に使用されるヒープの数を返します。
HeapsRelocatable	再配置可能なヒープの数を返します。
HttpPorts	Web サーバの TCP/IP ポート番号を返します。
HttpsPorts	Web サーバの TCP/IP ポート番号を返します。
IdleTimeout	デフォルトのアイドル・タイムアウトを返します。「-ti サーバ・オプション」 197 ページを参照してください。
IsEccAvailable	ECC DLL がインストールされている場合は Yes を返し、それ以外の場合は No を返します。
IsFipsAvailable	FIPS DLL がインストールされている場合は Yes を返し、それ以外の場合は No を返します。
IsIQ	サーバが IQ サーバである場合は Yes を返し、それ以外の場合は No を返します。
IsNetworkServer	ネットワーク・データベース・サーバに接続している場合は Yes を返し、パーソナル・データベース・サーバに接続している場合は No を返します。
IsRsaAvailable	RSA DLL がインストールされている場合は Yes を返し、それ以外の場合は No を返します。
IsRuntimeServer	機能が制限されたデスクトップ・ランタイム・データベース・サーバに接続している場合は Yes を返し、それ以外の場合は No を返します。
Language	サーバのロケール言語を返します。
LastConnectionProperty	最後の接続プロパティを表す番号を返します。
LastDatabaseProperty	最後のデータベース・プロパティを表す番号を返します。
LastOption	データベース・オプションに対応する最後の接続プロパティを表す番号を返します。
LastServerProperty	最後のサーバ・プロパティを表す番号を返します。
LegalCopyright	ソフトウェアの著作権を示す文字列を返します。
LegalTrademarks	ソフトウェアの商標を示す文字列を返します。
LicenseCount	ライセンスされたシートまたはプロセッサの数を返します。?
LicensedCompany	ライセンスされた会社の名前を返します。
LicensedUser	ライセンスされたユーザの名前を返します。

プロパティ	説明
LicenseType	ライセンス・タイプを返します。ネットワーク・シート (per-seat) または CPU ベースのいずれかです。
LivenessTimeout	クライアント活性タイムアウトのデフォルトを返します。「-tl サーバ・オプション」 197 ページを参照してください。
LockedCursorPages	メモリ内でカーソル・ヒープを固定するために使用されているページ数を返します。
LockedHeapPages	キャッシュ内でロックされているヒープ・ページの数を返します。
MachineName	データベース・サーバを実行しているコンピュータの名前を返します。通常は、コンピュータのホスト名です。
MainHeapBytes	グローバル・サーバ・データ構造に使用されているバイト数を返します。
MainHeapPages	グローバル・サーバ・データ構造に使用されているページ数を返します。
MapPhysicalMemoryEng	Address Windowing Extensions を使用して、データベース・ページのアドレス領域がキャッシュ内の物理メモリにマップされている頻度を返します。
MaxCacheSize	許容最大キャッシュ・サイズ (キロバイト単位) を返します。
MaxConnections	サーバが許容する同時接続の最大数を返します。デフォルト値は、パーソナル・サーバの場合は 10、ネットワーク・サーバの場合は約 32000 です。この値を小さくするには、-gm サーバ・オプションを使用します。「-gm サーバ・オプション」 171 ページを参照してください。 通常は、コンピュータ・リソースの制約から、ネットワーク・サーバへの接続の最大数はデフォルト値より小さくなります。
MaxMessage	サーバ・メッセージ・ウィンドウから取得できる行番号の現在の最大値を返します。この値は、サーバ・メッセージ・ウィンドウに最後に表示されたメッセージを表します。
Message, *linenumber*	サーバ・メッセージ・ウィンドウから取得したメッセージ行を返します。行の先頭にはメッセージが表示された日付と時刻が追加されています。2 番目のパラメータは行番号を指定します。 PROPERTY("message") によって返された値が、サーバのメッセージ・ウィンドウに書き込まれた出力の最初の行です。PROPERTY("message", i) を呼び出すと、サーバ出力の i 行目 (ゼロを最初の行とする) が返されます。バッファは有限であるため、メッセージの生成とともに最初の行が削除されていき、メモリからなくなる場合があります。この場合は、NULL が返されます。

プロパティ	説明
MessageText, *linenumber*	サーバ・メッセージ・ウィンドウ内の指定された行番号に対応するテキストを返します。このテキストの先頭には日付と時刻は追加されていません。2 番目のパラメータは行番号を指定します。
MessageTime, *linenumber*	サーバ・メッセージ・ウィンドウ内の指定された行番号に対応する日付と時刻を返します。2 番目のパラメータは行番号を指定します。
MessageWindowSize	サーバ・メッセージ・ウィンドウから取得できる行の最大数を返します。
MinCacheSize	許容最小キャッシュ・サイズ (キロバイト単位) を返します。
MultiPacketsReceived	クライアント／サーバ通信中に受信したマルチパケット要求の数を返します。
MultiPacketsSent	クライアント／サーバ通信中に送信されたマルチパケット応答の数を返します。
MultiPageAllocs	マルチページ・キャッシュ割り付けの数を返します。
MultiProgrammingLevel	サーバが一度に処理できる同時タスクの最大数を返します。この値を超える数の同時タスクが存在する場合、要求はキューイングされます。この動作を変更するには、-gn サーバ・オプションを使用します。「-gn サーバ・オプション」 172 ページを参照してください。
Name	データベースへの接続に使用するサーバの別名を返す (別名が指定されている場合)。別名が指定されていない場合は、実際のサーバ名を返します。「-sn オプション」 225 ページを参照してください。
NativeProcessorArchitecture	プロセッサをエミュレートできるプラットフォーム (X86 や Win64 など) のネイティブ・プロセッサ・タイプを表す文字列を返します。これ以外の場合はすべて、プロパティと同じ値 ('ProcessorArchitecture') を返します。 値には、次のものがあります。 ◆ 32 ビット版 Windows (Windows CE 以外) − X86 ◆ Windows CE − ARM ◆ 64 ビット版 Windows − IA64 または X86_64 ◆ NetWare − X86 ◆ Solaris − SPARC、X86、または X86_64 ◆ AIX − PPC ◆ MAC OS − PPC ◆ HP − PA_RISC または IA64 ◆ Linux − X86、IA64、または X86_64 サポートされているプラットフォームの完全なリストについては、「SQL Anywhere がサポートするプラットフォームおよびエンジニアリング・サポート状況」を参照してください。

プロパティ	説明
NumLogicalProcessors	サーバ・コンピュータ上の論理プロセッサ数 (コアとハイパースレッドを含む) を返します。
NumLogicalProcessorsUsed	データベース・サーバが使用する論理プロセッサの数を返します。Windows の場合、使用する論理プロセッサ数を変更するには、-gtc オプションを使用します。「-gtc サーバ・オプション」 175 ページを参照してください。
NumPhysicalProcessors	サーバ・コンピュータで有効となっている物理プロセッサの数を返します。この値は、NumLogicalProcessors の値を物理プロセッサあたりのコア数またはハイパースレッド数で割った数に相当します。Windows 以外のプラットフォームでは、コアまたはハイパースレッドが物理プロセッサとして計数される場合があります。
NumPhysicalProcessorsUsed	データベース・サーバが使用する物理プロセッサの数を返します。パーソナル・サーバの場合、プラットフォームによってはプロセッサ数が 1 つに制限されることがあります。Windows でネットワーク・データベース・サーバが使用する物理プロセッサの数を変更するには、-gt オプションを使用します。「-gt サーバ・オプション」 174 ページを参照してください。
OmniIdentifier	このプロパティはシステムで使用するために予約されています。このオプションの設定は変更しないでください。
PacketsReceived	受信したクライアント／サーバ通信パケットの数を返します。
PacketsReceivedUncomp	圧縮が無効になっている場合にクライアント／サーバ通信中に受信したパケットの数を返します (この値は、圧縮が無効の場合は PacketsReceived の値と同じ)。
PacketsSent	送信されたクライアント／サーバ通信パケットの数を返します。
PacketsSentUncomp	圧縮が無効になっている場合にクライアント／サーバ通信中に送信されたパケットの数を返します (この値は、圧縮が無効の場合は PacketsSent の値と同じ)。
PageSize	データベース・サーバのキャッシュ・ページのサイズを返します。-gp オプションを使用して設定可能です。設定しない場合はコマンド・ラインで指定したデータベースの最大ページ・サイズです。
PeakCacheSize	現在のセッションでキャッシュが到達した最大値 (キロバイト単位) を返します。
Platform	ソフトウェアが実行されているオペレーティング・システムを返します。たとえば、Windows 2000 を実行中の場合、このプロパティは Windows2000 を返します。

プロパティ	説明
PlatformVer	ソフトウェアを実行しているオペレーティング・システムの名前を返します。この名前には、ビルド番号やサービス・パックなどの情報も含まれています。たとえば、 Windows 2000 Build 2195 Service Pack 3 などを返します。
ProcessCPU	サーバ・プロセスにおける CPU 使用率の統計値を返します。値は秒数で指定します。このプロパティは Windows と UNIX でサポートされています。Windows CE または NetWare ではサポートされていません。 このプロパティが返す値は、データベース・サーバが起動してからの累積値です。この値は Windows タスク マネージャや Windows パフォーマンス モニタなどのアプリケーションに表示される瞬時値とは一致しません。
ProcessCPUSystem	システムによるプロセス CPU の使用状況を返します。値は秒数で指定します。このプロパティは Windows と UNIX でサポートされています。Windows CE や NetWare ではサポートされていません。 このプロパティが返す値は、データベース・サーバが起動してからの累積値です。この値は Windows タスク マネージャやパフォーマンス モニタなどのアプリケーションに表示される瞬時値とは一致しません。
ProcessCPUUser	ユーザによるプロセス CPU の使用状況を返します。値は秒数で指定します。このプロパティは Windows と UNIX でサポートされています。Windows CE や NetWare ではサポートされていません。 このプロパティが返す値は、データベース・サーバが起動してからの累積値です。この値は Windows タスク マネージャやパフォーマンス モニタなどのアプリケーションに表示される瞬時値とは一致しません。
ProcessorArchitecture	プロセッサ・タイプを表す文字列を返します。値には、次のものがあります。 ◆ 32 ビット版 Windows (Windows CE 以外) − X86 ◆ 64-bit Windows − IA64 または X86_64 ◆ Windows CE − ARM ◆ NetWare − X86 ◆ Solaris − SPARC、X86、または X86_64 ◆ AIX − PPC ◆ MAC OS − PPC ◆ HP − PA_RISC または IA64 ◆ Linux − X86、IA64、または X86_64
ProductName	ソフトウェアの名前を返します。
ProductVersion	実行中のソフトウェアのバージョンを返します。

プロパティ	説明
ProfileFilterConn	特定の接続をプロファイルするプロシージャがオンの場合にモニタされる接続の ID を返します。それ以外の場合、空の文字列を返します。ユーザによるプロシージャのプロファイルは、sa_server_option プロシージャを使って制御します。「sa_server_option システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
ProfileFilterUser	特定のユーザのプロシージャ・プロファイルがオンの場合にモニタされるユーザの名前を返します。それ以外の場合、空の文字列を返します。ユーザによるプロシージャのプロファイルは、sa_server_option プロシージャを使って制御します。「sa_server_option システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
QueryHeapPages	クエリ処理 (ハッシュ操作とソート操作) に使用されるキャッシュ・ページの数を返します。
QuittingTime	サーバのシャットダウン時間を返します。シャットダウン時間が指定されていない場合は、none を返します。
RememberLastPlan	オプティマイザが返す最後のクエリ最適化プランをサーバが記録する場合は、Yes を返します。「-zp サーバ・オプション」 213 ページを参照してください。
RememberLastStatement	サーバが各接続で準備された最後の文を記録している場合は Yes を返し、それ以外の場合は No を返します。「-zl サーバ・オプション」 211 ページを参照してください。
RemoteputWait	リモート・プットの待機数を返します。
Req	サーバが新しい要求を処理するか、または既存の要求の処理を続行できる状態になった回数を返します。
RequestFilterConn	接続のロギング情報をフィルタしている場合は、その接続の ID を返し、それ以外の場合は -1 を返します。
RequestFilterDB	データベースのロギング情報をフィルタしている場合は、そのデータベースの ID を返し、それ以外の場合は -1 を返します。
RequestLogFile	要求ロギング・ファイルの名前を返します。要求がロギングされていない場合は、空の文字列を返します。「sa_server_option システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
RequestLogging	現在の要求ロギング設定を示す値として、SQL、PLAN、HOSTVARS、PROCEDURES、TRIGGERS、OTHER、BLOCKS、REPLACE、ALL、NONE のいずれかを返します。「sa_server_option システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
RequestLogMaxSize	要求ログ・ファイルの最大サイズを返します。「-zs サーバ・オプション」 215 ページを参照してください。

プロパティ	説明
RequestLogNumFiles	保管される要求ログ・ファイルの数を返します。「sa_server_option システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
RequestTiming	要求タイミングがオンになっている場合は Yes を返し、オフになっている場合は No を返します。要求タイミングをオンにするには、-zr データベース・サーバ・オプションを使用します。「-zt オプション」 216 ページを参照してください。
RequestsReceived	クライアント／サーバ通信要求またはラウンド・トリップの数を返します。このプロパティでは、PacketsReceived とは異なり、マルチパケット要求を 1 つの要求として数え、活性パケットを計数の対象から除外します。
SendFail	通信プロトコルがパケット送信に失敗した回数を返します。
ServerName	現在の接続におけるサーバの名前を返します。この値によって、稼働しているサーバのうちどれがデータベース・ミラーリング構成のプライマリ・サーバであるかを特定できます。「データベース・ミラーリングの概要」 888 ページを参照してください。
StartDBPermission	-gd サーバ・オプションの設定を返します。値は、DBA、all、none のいずれかです。「-gd サーバ・オプション」 168 ページを参照してください。
StartTime	サーバが起動された日付／時刻を返します。
TempDir	テンポラリ・ファイルが格納されているサーバ上のディレクトリを返します。
TimeZoneAdjustment	サーバのローカル時間を表示するために協定世界時 (UTC: Coordinated Universal Time) に加算する必要がある時間 (分単位) を返します。
TotalBuffers	ネットワーク・バッファの総数を返します。
UniqueClientAddresses	ネットワーク・サーバに接続しているユニークなクライアント・ネットワーク・アドレスの数を返します。
UnschReq	使用できるサーバ・スレッドの解放を待ってキューイングされている要求の数を返します。

データベース・レベルのプロパティ

次の表は、サーバ上の各データベースに必要なプロパティを示します。

例

- **データベース・プロパティの値を取り出すには、次の手順に従います。**

  DB_PROPERTY システム関数を使用します。たとえば、次の文は、現在のデータベースのページ・サイズを返します。

  ```
  SELECT DB_PROPERTY ( 'PageSize' );
  ```

- **すべてのデータベース・プロパティの値を取り出すには、次の手順に従います。**

  sa_db_properties システム・プロシージャを使用します。

  ```
  CALL sa_db_properties;
  ```

参照

- 「DB_EXTENDED_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』
- 「サーバ・レベルのプロパティ」 540 ページ
- 「接続レベルのプロパティ」 518 ページ

説明

プロパティ	説明
AccentSensitive	アクセントの区別のステータスを返します。データベースでアクセントが区別される場合は Yes、区別されない場合は No、フランス語のアクセントの区別が適用されている場合は FRENCH を返します。
Alias	データベース名を返します。
AlternateServerName	データベースに関連付けられているサーバの別名を返す (別名が指定されている場合)。「-sn オプション」 225 ページを参照してください。
ArbiterState	次のいずれかの値を返します。 ♦ **NULL** ミラーリングされていないデータベースに接続している。 ♦ **connected** 監視サーバはプライマリ・サーバに接続している。 ♦ **disconnected** 監視サーバはプライマリ・サーバに接続していない。 「データベース・ミラーリングの概要」 888 ページを参照してください。
AuditingTypes	現在有効な監査のタイプを返します。「auditing オプション [データベース]」 428 ページを参照してください。

プロパティ	説明
BlankPadding	データベースでブランク埋め込み機能が有効になっている場合は、On を返します。それ以外の場合は、Off を返します。
CacheHits	キャッシュ内の検索において一致したデータベース・ページの数を返します。
CacheRead	キャッシュの中で検索されたデータベース・ページの数。
CacheReadIndInt	キャッシュから読み込まれたインデックス内部ノード・ページの数を返します。
CacheReadIndLeaf	キャッシュから読み込まれたインデックス・リーフ・ページの数を返します。
CacheReadTable	キャッシュから読み込まれたテーブル・ページの数を返します。
Capabilities	データベースに対して有効な機能ビットを返します。このプロパティは、主にテクニカル・サポートを行うために使用されます。
CaseSensitive	大文字と小文字の区別のステータスを返します。データベースで大文字と小文字が区別される場合は、On を返します。それ以外の場合は、Off を返します。大文字と小文字が区別されるデータベースでは、データの比較において大文字と小文字が異なるものとして扱われます。この設定は識別子における大文字と小文字の区別には影響しません。パスワードについては、常に大文字と小文字が区別されます。「大文字と小文字の区別」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
CatalogCollation	カタログに使用される照合の識別子を返します。このプロパティには、プロパティ値を問い合わせるときに指定できる拡張機能があります。「DB_EXTENDED_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
CharSet	データベースの CHAR 文字セットを返します。 このプロパティには、プロパティ値を問い合わせるときに指定できる拡張機能があります。「DB_EXTENDED_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
CheckpointLogBitmapPagesWritten	チェックポイント・ログのビットマップへの書き込み数を返します。
CheckpointLogBitmapSize	チェックポイント・ログのビットマップ・サイズを返します。
CheckpointLogCommitToDisk	チェックポイント・ログのディスクへのコミット回数を返します。
CheckpointLogPageInUse	使用中のチェックポイント・ログ・ページ数を返します。

プロパティ	説明
CheckpointLogPagesRelocated	再配置されたチェックポイント・ログ・ページ数を返します。
CheckpointLogPagesWritten	書き込みが完了したチェックポイント・ログ・ページ数を返します。
CheckpointLogSavePreimage	データベース・ページのプレイメージがチェックポイント・ログに追加されている頻度を返します。
CheckpointLogSize	チェックポイント・ログのサイズ (ページ数) を返します。
CheckpointLogWrites	チェックポイント・ログへの書き込み数を返します。
CheckpointUrgency	最後のチェックポイントからの経過時間を返します。この値は、データベースのチェックポイント時間の設定に対するパーセンテージで表されます。
Checksum	データベース・ページのチェックサムが有効になっている場合は、On を返します。それ以外の場合は、Off を返します。重要なページには必ずチェックサムが含まれています。
Chkpt	実行されたチェックポイントの数を返します。
ChkptFlush	チェックポイント実行中に書き出された一連のページの数を返します。
ChkptPage	トランザクション・ログ・チェックポイントの数を返します。
CleanablePagesAdded	データベース・サーバの起動後、クリアされるようにマークが付けられたページの数を返します。
CleanablePagesCleaned	データベース・サーバの起動後、クリアされたデータベース・ページの数を返します。
CleanableRowsAdded	データベース・サーバの起動後、削除されるようにマークが付けられたローの数を返します。
CleanableRowsCleaned	データベース・サーバの起動後、削除されたシャドー・テーブルのローの数を返します。
Collation	データベースが使用する照合を返します。使用可能な照合のリストについては、「サポートされている照合と代替照合」 360 ページを参照してください。 このプロパティには、プロパティ値を問い合わせるときに指定できる拡張機能があります。「DB_EXTENDED_PROPERTY 関数 [システム]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
CommitFile	ディスク・キャッシュのフラッシュをサーバが強制的に実行した回数を返します。Windows と NetWare では、バッファなし (ダイレクト) IO を使用している場合、ディスク・キャッシュのフラッシュは不要です。

プロパティ	説明
ConnCount	データベースとの接続の数を返します。
ConnsDisabled	現在のデータベースとの接続が無効である場合は On を返し、それ以外の場合は Off を返します。
CurrentRedoPos	次のデータベース操作がロギングされるトランザクション・ログ・ファイルの現在のオフセットを返します。
CurrIO	サーバが発行し、まだ完了していない現在のファイル I/O の数を返します。
CurrRead	サーバが発行し、まだ完了していない現在のファイル読み込み数を返します。
CurrWrite	サーバが発行し、まだ完了していない現在のファイル書き込み数を返します。
DatabaseCleaner	データベース・クリーナが有効になっているかどうかを示す On または Off を返します。
DBFileFragments	データベース・ファイルのフラグメント数を返します。このプロパティは Windows でサポートされています。
DiskRead	ディスクから読み込まれたページの数を返します。
DiskReadIndInt	ディスクから読み込まれたインデックス内部ノード・ページの数を返します。
DiskReadIndLeaf	ディスクから読み込まれたインデックス・リーフ・ページの数を返します。
DiskReadTable	ディスクから読み込まれたテーブル・ページの数を返します。
DiskWrite	ディスクに書き込まれた修正ページの数を返します。
DriveType	データベース・ファイルが配置されているドライブのタイプを返します。値は、CD、FIXED、RAMDISK、REMOTE、REMOVABLE、UNKNOWN のいずれかです。 UNIX では、UNIX のバージョンとドライブのタイプにより、ドライブ・タイプを判別できない場合があります。そのような場合、「UNKNOWN」が返されます。 このプロパティには、プロパティ値を問い合わせるときに指定できる拡張機能があります。「DB_EXTENDED_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
Encryption	データベース暗号化またはテーブル暗号化のタイプを返します。値は、None、Simple、AES、AES_FIPS のいずれかです。

プロパティ	説明
EncryptionScope	データベース内に暗号化が可能な部分がある場合、その部分を示す値を返します。値は、TABLE、DATABASE、NONE のいずれかです。 TABLE は、テーブルの暗号化が有効になっていることを示します。DATABASE は、データベース全体が暗号化されることを示します。NONE は、テーブルの暗号化が有効になっておらず、データベースが暗号化されないことを示します。
ExprCacheAbandons	ヒット率が低すぎるために式キャッシュが完全に中止された回数を返します。
ExprCacheDropsToReadOnly	ヒット率が低いために式キャッシュが読み込み専用ステータスに変更された回数を返します。
ExprCacheEvicts	式キャッシュからの出力数を返します。
ExprCacheHits	式キャッシュ内のヒット数を返します。
ExprCacheInserts	式キャッシュに挿入された値の数を返します。
ExprCacheLookups	式キャッシュ内で実行されたルックアップの回数を返します。
ExprCacheResumesOfReadWrite	ヒット率の上昇によって式キャッシュが読み込み／書き込みステータスに戻った回数を返します。
ExprCacheStarts	式キャッシュが開始された回数を返します。
ExtendDB	データベース・ファイルを拡張するために追加されたページの数を返します。
ExtendTempWrite	テンポラリ・ファイルを拡張するために追加されたページの数を返します。
File	パスを含むデータベース・ルート・ファイル名を返します。このプロパティには、プロパティ値を問い合わせるときに指定できる拡張機能があります。「DB_EXTENDED_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
FileSize	SYSTEM DB 領域のファイル・サイズ (ページ単位) を返します。このプロパティには、プロパティ値を問い合わせるときに指定できる拡張機能があります。「DB_EXTENDED_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
FreePages	SYSTEM DB 領域の空きページ数を返します。FreePages プロパティは、バージョン 8.0.0 以降で作成されたデータベースでのみサポートされます。 このプロパティには、プロパティ値を問い合わせるときに指定できる拡張機能があります。「DB_EXTENDED_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

プロパティ	説明
FullCompare	インデックスのハッシュ値を超えて実行された比較の回数を返します。
GetData	GETDATA 要求の数を返します。
GlobalDBID	レプリケーション環境でのユニークなプライマリ・キー値の生成に使用される global_database_id オプションの値を返します。
HashForcedPartitions	メモリ競合が原因でハッシュ演算子が分割された回数を返します。
HasCollationTailoring	データベースの作成時に照合の適合化が指定されていたかどうかを示す応答を返します。使用できる値は On と Off です。
HashRowsFiltered	プローブ・ローがビットベクトル・フィルタによって拒否される割合を返します。
HashRowsPartitioned	ローがハッシュ・ワーク・テーブルに書き込まれる割合を返します。
HashWorkTables	ハッシュベースの演算用に作成されたワーク・テーブルの数を返します。
IdentitySignature	予約。
IdleCheck	サーバのアイドル・スレッドがアクティブになり、アイドル書き込み、アイドル・チェックポイントなどを実行した回数を返します。
IdleChkpt	サーバのアイドル・スレッドが完了したチェックポイントの数を返します。アイドル・スレッドが最後のダーティ・ページをキャッシュに書き出すたびに、アイドル・チェックポイントが発生します。
IdleChkTime	アイドル I/O 中にチェックポイントに費やした 100 分の 1 秒単位の時間を返します。
IdleWrite	サーバのアイドル・スレッドによるディスク書き込みの数を返します。
IndAdd	インデックスに追加されたエントリの数を返します。
IndLookup	インデックス内で検索されたエントリの数を返します。
IOParallelism	DB 領域がサポートする同時 I/O 操作の推定回数を返します。このプロパティには、プロパティ値を問い合わせるときに指定できる拡張機能があります。「DB_EXTENDED_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
IOToRecover	データベースのリカバリに必要な I/O 操作の推定回数を返します。

プロパティ	説明
IQStore	IQ ストアがオンであるかオフであるかを示す値を返します。このプロパティは、SQL Anywhere については常に Off を返します。
JavaVM	データベース・サーバがデータベース内で Java を実行するために使用する Java VM を返します。
Language	データベース照合によってサポートされている言語のカンマで区切ったリストを返します。言語は 2 文字の ISO フォーマットです。言語が不明である場合、このプロパティは NULL を返します。2 文字の ISO フォーマット言語名とそれに対応する言語のリストについては、「ロケール言語の知識」 346 ページを参照してください。
LockCount	接続で保持されているロックの数を返します。
LockTablePages	ロック情報の保持に使用されているページの数を返します。
LogFileFragments	ログ・ファイルのフラグメント数を返します。このプロパティは Windows でサポートされています。
LogFreeCommit	Redo Free Commit の数を返します。Redo Free Commit が起こるのは、トランザクション・ログのコミットが要求されているが、ログはすでに書き込まれている (コミットはいつでも可能) ときです。
LogMirrorName	パスを含むトランザクション・ログ・ミラーのファイル名を返します。
LogName	パスを含むトランザクション・ログ・ファイル名を返します。
LogWrite	トランザクション・ログに書き込まれたページの数を返します。
LTMGeneration	LTM または Replication Agent の世代番号を返します。このプロパティは、主にテクニカル・サポートを行うために使用されます。
LTMTrunc	Replication Agent 用に最後に確認されたログ・オフセットを返します。
MapPages	ロック・テーブル、頻度テーブル、テーブル・レイアウトへのアクセスに使用されたマップ・ページの数を返します。
MaxIO	CurrIO が到達した最大値を返します。
MaxRead	CurrRead が到達した最大値を返します。
MaxWrite	CurrWrite が到達した最大値を返します。

プロパティ	説明
MirrorState	次のいずれかの値を返します。 ♦ **null** ミラーリングされていないデータベースに接続している。 ♦ **synchronizing** ミラー・サーバが接続されていないか、プライマリ・サーバのログ・ページの読み込みが完了していない。同期実行モードが非同期である場合にも、この値が返されます。 ♦ **synchronized** ミラー・サーバが接続され、プライマリ・サーバ上でコミットされたすべての変更が反映されている。 「データベース・ミラーリングの概要」 888 ページを参照してください。
MultiByteCharSet	データベースでマルチバイト文字セットが使用されている場合は、On を返します。それ以外の場合は、Off を返します。
Name	データベース名を返します (Alias と同じ機能)。
NcharCharSet	データベースの NCHAR 文字セットを返します。
NcharCollation	NCHAR データに使用される照合の名前を返します。このプロパティには、プロパティ値を問い合わせるときに指定できる拡張機能があります。「DB_EXTENDED_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
NextScheduleTime	指定したイベントの次の予定実行時間を返します。このプロパティに対するクエリには、DB_EXTENDED_PROPERTY 関数を使用します。「DB_EXTENDED_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
PageRelocations	テンポラリ・ファイルから読み込まれた再配置可能なヒープ・ページの数を返します。
PageSize	データベースのページ・サイズ (バイト単位) を返します。
PartnerState	次のいずれかの値を返します。 ♦ **NULL** ミラーリングされていないデータベースに接続している。 ♦ **connected** ミラー・サーバはプライマリ・サーバに接続している。 ♦ **disconnected** ミラー・サーバはプライマリ・サーバに接続されていない。 「データベース・ミラーリングの概要」 888 ページを参照してください。

プロパティ	説明
PreserveSource	On を返します。このプロパティは使用されなくなりました。
ProcedurePages	プロシージャで使用された再配置可能なヒープ・ページの数を返します。
ProcedureProfiling	データベースのプロシージャ・プロファイリングが有効になっている場合は、On を返します。それ以外の場合は、Off を返します。
QueryBypassed	プラン・キャッシュから再利用された要求の数を返します。
QueryCachedPlans	すべての接続においてキャッシュされている実行プランの数を返します。
QueryCachePages	実行プランの保存に使用されているキャッシュ・ページの数を返します。
QueryJHToJNLOptUsed	ハッシュ・ジョインがネスト・ループ・ジョインに変換された回数を返します。
QueryLowMemoryStrategy	メモリ不足が原因でサーバが実行プランを実行中に変更した回数を返します。使用できるメモリがオプティマイザの推定よりも少ない、または実行プランが必要とするメモリがオプティマイザの推定よりも多い場合に、プランが変更されることがあります。
QueryOptimized	完全に最適化された要求の数を返します。
QueryRowsBufferFetch	バッファを使用してフェッチされたローの数を返します。
QueryRowsMaterialized	クエリ処理中にローがワーク・テーブルに書き込まれる割合を返します。
ReadOnly	データベースが読み込み専用モードで実行されている場合は、On を返します。それ以外の場合は、Off を返します。
ReceivingTracingFrom	トレーシング・データが保存されていたデータベースの名前を返します。トレーシングは追加されていない場合は、ブランク文字列を返します。
RecoveryUrgency	データベースのリカバリに要する推定時間を返します。この値は、データベースのリカバリ時間設定に対する割合で表されます。「-gr サーバ・オプション」 173 ページを参照してください。
RecursiveIterations	再帰ユニオンの反復回数を返します。
RecursiveIterationsHash	再帰ハッシュ・ジョインでハッシュ方式が使用された回数を返します。
RecursiveIterationsNested	再帰ハッシュ・ジョインでネスト・ループ方式が使用された回数を返します。

プロパティ	説明
RecursiveJNLMisses	再帰ハッシュ・ジョインでのインデックス・プローブ・キャッシュ・ミスの数を返します。
RecursiveJNLProbes	再帰ハッシュ・ジョインにおけるインデックス・プローブの試行回数を返します。
RelocatableHeapPages	再配置可能なヒープ (カーソル、文、プロシージャ、トリガ、ビューなど) で使用されるページの数を返します。
RemoteTrunc	SQL Remote Message Agent 用に最後に確認されたログ・オフセットを返します。
RollbackLogPages	ロールバック・ログのページ数を返します。
SendingTracingTo	トレーシング・データの送信先を示す接続文字列を返します。トレーシングは追加されていない場合は、ブランク文字列を返します。
SnapshotCount	接続に関連付けられているスナップショットの数を返します。
SnapshotIsolationState	次のいずれかの値を返します。 ◆ **On** データベースでスナップショット・アイソレーションが有効になっている。 ◆ **Off** データベースでスナップショット・アイソレーションが無効になっている。 ◆ **in_transition_to_on** 現在のトランザクションの完了後にスナップショット・アイソレーションが有効となる。 ◆ **in_transition_to_off** 現在のトランザクションの完了後にスナップショット・アイソレーションが無効となる。 「allow_snapshot_isolation オプション [データベース]」 422 ページを参照してください。
SortMergePasses	ソート中に使用されたマージ・パスの数を返します。
SortRowsMaterialized	ローがソート・ワーク・テーブルに書き込まれる割合を返します。
SortRunsWritten	ソート中に書き込まれたソート実行の数を返します。
SortSortedRuns	実行の生成中に作成されたソート実行の数を返します。
SortWorkTables	ソート用に作成されたワーク・テーブルの数を返します。
SyncTrunc	Mobile Link クライアントの dbmlsync 実行プログラム用に最後に確認されたログ・オフセットを返します。
TempFileName	パスを含むデータベース・テンポラリ・ファイル名を返します。

プロパティ	説明
TempTablePages	テンポラリ・テーブルで使用されるテンポラリ・ファイルのページ数を返します。
TriggerPages	トリガで使用される再配置可能なヒープ・ページの数を返します。
UniqueIdentifier	On を返します。このプロパティは使用されなくなりました。
VersionStorePages	スナップショット・アイソレーションが有効な場合にロー・バージョン・ストアで使用されるテンポラリ・ファイルのページ数を返します。
ViewPages	ビューで使用される再配置可能なヒープ・ページの数を返します。
XPathCompiles	データベース・サーバの起動後、(openxml プロシージャ を使用する) XPath クエリがコンパイルされた回数を返します。

第 14 章

Adaptive Server Anywhere の制限

目次

SQL Anywhere のサイズと数の制限

SQL Anywhere データベースにおけるオブジェクトのサイズと数の制限について次の表に示します。実際には、コンピュータのメモリ、CPU、ディスク容量から受ける制限の方が厳しいのが普通です。

項目	制限
データベース・サイズ	13 ファイル／データベース。オペレーティング・システムとファイル・システムが許可する各ファイルの最大サイズ。
DB 領域サイズ	2^{28} × ページ・サイズ
テンポラリ・ファイル・サイズ	2^{28} × ページ・サイズ
フィールドの大きさ	2 GB
ファイル・サイズ (FAT 12)	16 MB
ファイル・サイズ (FAT 16)	2 GB
ファイル・サイズ (FAT 32)	4 GB
ファイル・サイズ (NTFS、NetWare (NSS ボリューム)、HP-UX 11.0 以降、Solaris 2.6 以降、Linux 2.4 以降)	◆ 512 GB (2 KB ページに対して) ◆ 1 TB (4 KB ページに対して) ◆ 2 TB (8 KB ページに対して)
NetWare (従来のボリューム)	4 GB
ファイル・サイズ (その他のプラットフォームとファイル・システム)	2 GB
最大キャッシュ・サイズ (非 AWE キャッシュ) (Windows 2000 Professional、Windows 2000 Server、Windows XP Home Edition、Windows XP Professional、Windows Server 2003 Web Edition、Windows Server 2003 Standard Edition)	1.8 GB
最大キャッシュ・サイズ (非 AWE キャッシュ) (Windows 2000 Advanced Server、Windows 2000 Enterprise Server、Windows 2000 Datacenter Server、Windows Server 2003 Enterprise Edition、Windows Server 2003 Datacenter Edition、Windows Vista Ultimate、Windows Vista Enterprise、Windows Vista Business、Windows Vista Home Premium、Windows Vista Home Basic)	2.7 GB

項目	制限
最大キャッシュ・サイズ (AWE キャッシュ) (Windows 2000 Professional、Windows 2000 Server、Windows 2000 Advanced Server、Windows 2000 Datacenter Server、Windows XP Home Edition、Windows XP Professional、Windows Server 2003 Web Edition、Windows Server 2003 Standard Edition、Windows Server 2003 Enterprise Edition、Windows Server 2003 Datacenter Edition)	使用可能な全メモリの 100% ー 128 MB
最大キャッシュ・サイズ (Windows CE)	デバイス上の使用可能メモリによる
最大キャッシュ・サイズ (UNIX−Solaris、x86 Linux、AIX、HP)	2 GB (32 ビット・サーバに対して)
最大キャッシュ・サイズ (Win 64)	64 ビット・サーバの物理メモリによる
最大キャッシュ・サイズ (UNIX−Itanium Linux、Itanium HP-UX)	64 ビット・サーバの物理メモリによる
最大キャッシュ・サイズ (NetWare)	2 GB
最大インデックス・エントリ・サイズ	制限なし
データベース数／サーバ	255
カラム数／テーブル	◆ $((\text{ページ・サイズ})/4)^2$ (32 ビット・サーバに対して) ◆ $((\text{ページ・サイズ})/8)^2$ (64 ビット・サーバに対して) 注意：カラム数を過度に多くすることは可能だがパフォーマンスに影響する。
インデックス数／テーブル	2^{32}
ロー数／データベース	$4096 \times 2^{28} \times 13$
ロー数／テーブル	4096×2^{28}
テーブル数／データベース	$2^{32} - 2^{20} - 1 = 4293918719$
テンポラリ・テーブル数／接続	$2^{20} = 1048576$
参照されるテーブル数／トランザクション	制限なし
ストアド・プロシージャ数／データベース	$2^{32} - 1 = 4294967295$
同時実行文の数／データベース・サーバ	20 × *データベース接続数*+65534
イベント数／データベース	$2^{31} - 1 = 2147483647$
トリガ数／データベース	$2^{32} - 1 = 4294967295$

項目	制限
ロー・サイズ	ファイル・サイズにより制限
テーブルの大きさ	最大ファイル・サイズ。テーブルのユーザ定義インデックスは、テーブルとは別に保存可能。
文字列	2 GB
バイナリ・データ型	2 GB
識別子 (ユーザ ID、テーブル名、カラム名を含む)	128 バイト
データベース・サーバ名	◆ **TCP/IP** 250 バイト ◆ **共有メモリ** 250 バイト ◆ **SPX** 32 バイト 「-n サーバ・オプション」 181 ページと「EngineName 接続パラメータ [ENG]」 249 ページを参照してください。
DEFAULT 式または COMPUTE 式 (ALTER TABLE 文などで使用)	ページ・サイズにより制限 (ページ・サイズ - 64 バイト)

パート III. データベースの管理

パート III では、SQL Anywhere に付属のツールを使用してデータベースを管理する方法について説明します。

第 15 章

SQL Anywhere 管理ツール

目次

Sybase Central

Sybase Central は、データベースと関連製品を管理するためのグラフィカル・ツールです。サーバやデータベース、およびそれらに含まれているオブジェクトの管理に役立ちます。

Sybase Central で [ヘルプ] – [Sybase Central] を選択すると、Sybase Central の使用や設定についての詳細情報を参照できます。

Sybase Central の主要な機能

- **簡単なコマンド・アクセス**　オブジェクトを選択すると Sybase Central の [ファイル] メニューが自動的に更新され、そのオブジェクトに直接関連するコマンドが表示されます。オブジェクトを右クリックすると表示されるポップアップ・メニューからも、コマンドにアクセスできます。
- **タスク・ウィザード**　Sybase Central には、新しいオブジェクトを追加するための、段階を踏んでタスクを実行するウィザードが準備されています。
- **ドラッグ・アンド・ドロップ機能**　Sybase Central では、多くのオペレーションでドラッグ・アンド・ドロップ機能を利用できます。たとえば、異なるデータベースへテーブルをコピーするには、テーブルをクリックしてコピー先にドラッグするだけで実行できます。
- **キーボード・ショートカット**　一般的に使用される多くのコマンドにはキーボード・ショートカットがあります。ショートカットは、メニューのコマンド名の横に表示されています。「Sybase Central キーボード・ショートカット」 576 ページを参照してください。
- **プラグイン・サポート**　プラグインを使用して、さまざまなデータベース製品やツールを管理できます。各プラグインには、このオンライン・ヘルプとは別に固有のオンライン・ヘルプが用意されています。

プラグイン

製品は、それぞれ異なるプラグインで管理されています。Sybase Central で製品を使用する前に、該当するプラグインを登録し、ロードする必要があります。プラグインは、製品のインストール時に自動的に登録され、ロードされます。

SQL Anywhere 10 には、次の製品に対応する Sybase Central プラグインが備えられています。

- SQL Anywhere データベース
- Ultra Light データベース
- Mobile Link 同期
- QAnywhere メッセージ

プラグイン・ファイルは、SQL Anywhere インスールの以下のロケーションに格納されます。

プラグイン	ファイル名とロケーション
SQL Anywhere 10	*install-dir¥java¥saplugin.jar*
Mobile Link 10	*install-dir¥java¥mlplugin.jar*
Ultra Light 10	*install-dir¥ultralite¥java¥ulplugin.jar*

プラグイン	ファイル名とロケーション
QAnywhere 10	*install-dir¥java¥qaplugin.jar*

SQL Anywhere 10 に含まれているプラグインを使用する方法の詳細については、次の項を参照してください。

- SQL Anywhere：「SQL Anywhere プラグインの使用」 581 ページ
- Mobile Link：「Mobile Link のモデル」 『Mobile Link - クイック・スタート』
- QAnywhere：「QAnywhere プラグイン」 『QAnywhere』
- Ultra Light：「Sybase Central からの Ultra Light データベースの作成」 『Ultra Light - データベース管理とリファレンス』と「Ultra Light データベースの操作」 『Ultra Light - データベース管理とリファレンス』

ライセンス契約に従って、Sybase Central を含む一連の管理ツールを配備できます。

アプリケーションとともに Sybase Central を配備する方法の詳細については、「管理ツールの配備」 『SQL Anywhere サーバ - プログラミング』を参照してください。

Sybase Central の起動

この項では、Sybase Central を起動し、Windows や UNIX 上のサンプル・データベースに接続する手順について説明します。

♦ Sybase Central を起動してサンプル・データベースに接続するには、次の手順に従います (Windows の場合)。

1. [スタート] − [プログラム] − [SQL Anywhere 10] − [Sybase Central] を選択します。

   Sybase Central が開きます。

2. [Sybase Central へようこそ] ダイアログで、[スキーマの表示および編集、またはデータベースの管理の実行] をクリックします。

   [Sybase Central へようこそ] ダイアログが表示されない場合は、[接続] − [SQL Anywhere 10 に接続] を選択します。

   [接続] ダイアログが表示されます。

3. [ID] タブで、[ODBC データ・ソース名] を選択し、**SQL Anywhere 10 Demo** と入力します。

4. [OK] をクリックして接続します。

Linux のデスクトップ・アイコンをサポートするバージョンの Linux を使用している場合で、SQL Anywhere 10 をインストールするときにこれらのアイコンをインストールするように選択したときは、次の手順を使用できます。

♦ Sybase Central を起動してサンプル・データベースに接続するには、次の手順に従います (Linux デスクトップ・アイコンの場合)。

1. デスクトップで [SQL Anywhere 10] フォルダを開きます。
2. Sybase Central をダブルクリックします。

   Sybase Central が開きます。
3. [Sybase Central へようこそ] ダイアログで、[スキーマの表示および編集、またはデータベースの管理の実行] をクリックします。

   [Sybase Central へようこそ] ダイアログが表示されない場合は、[接続] – [SQL Anywhere 10 に接続] を選択します。

   [接続] ダイアログが表示されます。
4. [ID] タブで、[ODBC データ・ソース名] を選択し、**SQL Anywhere 10 Demo** と入力します。

注意
以降の手順は、SQL Anywhere ユーティリティのソースを指定済みであることを前提としています。「UNIX と Mac OS X での環境変数の設定」 306 ページを参照してください。

♦ Sybase Central を起動してサンプル・データベースに接続するには、次の手順に従います (UNIX コマンド・ラインの場合)。

1. ターミナル・セッションで次のコマンドを入力します。

   ```
   scjview
   ```

   Sybase Central が開きます。
2. [Sybase Central へようこそ] ダイアログで、[スキーマの表示および編集、またはデータベースの管理の実行] をクリックします。

   [Sybase Central へようこそ] ダイアログが表示されない場合は、[接続] – [SQL Anywhere 10 に接続] を選択します。

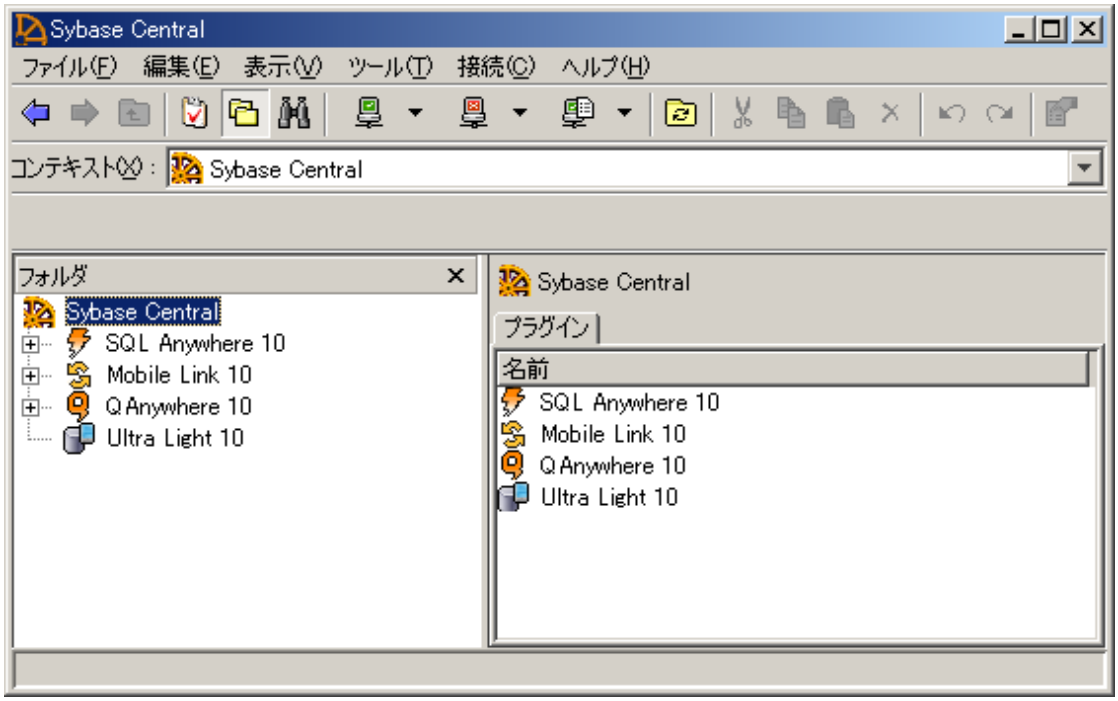
   [接続] ダイアログが表示されます。
3. [ID] タブで、[ODBC データ・ソース名] を選択し、**SQL Anywhere 10 Demo** と入力します。

Sybase Central のナビゲーション

この項では、Sybase Central のユーザ・インタフェースのナビゲーション方法について説明します。

Sybase Central のメイン・ウィンドウ

Sybase Central のメイン・ウィンドウ



ウィンドウ枠

Sybase Central のメイン・ウィンドウは左右に並んだ 2 つのウィンドウ枠に分割されています。

左ウィンドウ枠では、データベース・オブジェクトの階層ビューを表示するか、現在選択しているデータベース・オブジェクトのタスク・リストを表示することができます。

♦ 左ウィンドウ枠でタスク・リストまたはツリー・ビューを表示するには、次の手順に従います。

1. Sybase Central を起動します。
2. [ビュー] メニューから、[タスク] または [フォルダ] を選択し、タスク・リストまたはツリー・ビューを表示します。

右ウィンドウ枠には、左ウィンドウ枠で選択したコンテナの内容を表示するタブがあります。また、選択したコンテナについての情報も表示されます。両方のウィンドウ枠の表示は、[オプション] ダイアログ ([ツール] メニューからアクセス) で変更できます。

[ビュー] – [カラムの選択] を選択すると、右ウィンドウ枠のタブに表示されるカラムを設定できます。

ツールバー

メイン・ウィンドウのツールバーには、一般的なコマンドのボタンがあります。ツールバーの表示／非表示を切り替えるには、[ビュー] メニューの [ツールバー] を選択します。メイン・ツールバーでは、次の作業を実行できます。

- オブジェクト・ツリーのナビゲーション
- データベース、サーバ、製品プラグインへの接続または接続の解除
- [接続プロファイル] ダイアログにアクセス ([ツール] メニューからも可能)
- 現在のフォルダのビューの再表示
- オブジェクトの切り取り、コピー、貼り付け、または削除
- 変更の取り消しと再実行
- 選択したオブジェクトのプロパティ・シートの表示

ステータス・バー

メニュー間をナビゲーションすると、メイン・ウィンドウの下部に表示されるステータス・バーに、メニュー・コマンドの簡単な説明が表示されます。ステータス・バーを表示または非表示にするには、[ビュー] – [ステータス・バー] を選択します。

コンテキスト・ドロップダウン・リスト

コンテキスト・ドロップダウン・リストは、ツールバーの下に表示されるリストで、プラグインのオブジェクト・ツリーをナビゲーションできます。

接続プロファイルの使用

サーバまたはデータベースに最初に接続するときには、一般的にユーザ名やパスワードなどの接続パラメータを入力します。

Sybase Central からの次回以降の接続をより簡単に行うには、接続プロファイルを作成して、指定した名前で一連のパラメータを保存します。その後は、プロファイルを使用して 1 つの手順だけでオブジェクトに接続できます。

接続プロファイルを使用したり管理したりするには、[接続] – [接続プロファイル] を選択します。[接続プロファイル] ダイアログが表示され、以下の作業を実行できます。

- 新しい接続プロファイルの作成
- 接続プロファイルを使用した接続
- 既存のプロファイルの編集
- Sybase Central の起動時に自動的に接続するプロファイルの設定
- 接続プロファイルのインポートまたはエクスポート
- プロファイルの削除

注意
接続プロファイルは Sybase Central 固有です。ODBC アプリケーションを構築している場合は、ODBC データ・ソースを使用して接続プロファイルに似た機能を実現できます。「ODBC データ・ソースでの接続パラメータの保存」 63 ページを参照してください。

♦ 新しい接続プロファイルを作成するには、次の手順に従います。

1. Sybase Central から、[接続] – [接続プロファイル] を選択します。

   [接続プロファイル] ダイアログが表示されます。
2. [新規] をクリックします。
3. [新しい接続プロファイル] ダイアログで、新しいプロファイルの名前を入力し、適切なプラグインを選択します。このプラグインは、SQL Anywhere 10 または Mobile Link 10 などの製品です。
4. 必要に応じて、[この接続プロファイルを他のユーザと共有する] を選択し、他のユーザがプロファイルにアクセスできるようにします。これは UNIX などのマルチユーザ・プラットフォームで役立ちます。
5. [OK] をクリックします。

   [接続プロファイルの編集] ダイアログが表示されます。
6. [接続プロファイルの編集] ダイアログで、必要な値を入力し、[OK] をクリックしてダイアログを閉じます。

♦ Sybase Central の起動時に自動的に接続するには、次の手順に従います。

1. Sybase Central から、[接続] – [接続プロファイル] を選択します。

   [接続プロファイル] ダイアログが表示されます。
2. 接続プロファイルを選択します。
3. [スタートアップの設定] をクリックし、[起動時に使用] の設定を [はい] に変更します。

♦ 既存の接続プロファイルのパラメータを編集するには、次の手順に従います。

1. Sybase Central から、[接続] – [接続プロファイル] を選択します。

   [接続プロファイル] ダイアログが表示されます。
2. 接続プロファイルを選択します。
3. [編集] をクリックします。

   [接続プロファイルの編集] ダイアログが表示されます。

4. [接続プロファイルの編集] ダイアログで必要な値を編集します。

♦ 接続プロファイルをインポートするには、次の手順に従います。

1. Sybase Central から、[接続] – [接続プロファイル] を選択します。
   [接続プロファイル] ダイアログが表示されます。
2. [インポート] をクリックします。
   [接続プロファイル・ファイルのインポート] ダイアログが表示されます。
3. インポートする接続プロファイルを含むファイルの名前を指定します。
4. [OK] をクリックします。

♦ 接続プロファイルをエクスポートするには、次の手順に従います。

1. Sybase Central から、[接続] – [接続プロファイル] を選択します。
   [接続プロファイル] ダイアログが表示されます。
2. エクスポートする接続プロファイルを選択します。
3. [エクスポート] をクリックします。
   [名前を付けて保存] ダイアログが表示されます。
4. 接続プロファイルを保存するファイルの名前を入力します。
5. [保存] をクリックします。

Sybase Central キーボード・ショートカット

Sybase Central には、以下のキーボード・ショートカットが用意されています。

ファンクション・キー	説明
[Alt + Enter]	選択した項目の [プロパティ] ダイアログを開く
[Ctrl + C]	選択した内容をクリップボードにコピーする
[Ctrl + V]	クリップボードの内容を挿入する
[Ctrl + X]	選択した内容を切り取ってクリップボードに移動する
[Delete]	選択されている内容を削除する
[F1]	Sybase Central のヘルプを開く
[F5]	選択したフォルダの内容を再表示する

ファンクション・キー	説明
[F9]	[接続プロファイル] ダイアログを開く
[F11]	複数のプラグインがロードされている場合は [接続] メニューを開く。ロードされているプラグインが 1 つである場合は、そのプラグインの [接続] ダイアログを開きます。
[F12]	Sybase Central の接続が 1 つだけの場合は切断する。接続が複数の場合は、[F12] キーを押すと [切断] ダイアログが開き、切断する接続を選択できます。
[Shift + F10]	ポップアップ・メニューを開く

コード・エディタの使用

コード・エディタは、Sybase Central の右ウィンドウ枠に [SQL] タブとして表示されます。Sybase Central に別のウィンドウとして表示されたり、Interactive SQL に [SQL 文] ウィンドウ枠として表示されることもあります。コード・エディタを使用すると、コードやメッセージの表示、編集、印刷を行うことができます。

[SQL 文] ウィンドウ枠の使用については、「Interactive SQL メイン・ウィンドウの説明」 587 ページと「Interactive SQL キーボード・ショートカット」 604 ページを参照してください。

コード・エディタには、標準のテキスト編集機能に加えて次のような機能があります。

- ツールバーとステータス・バー
- 構文の自動強調表示
- 言語依存インデント
- テキストの検索と置換機能
- ファイルを開いたり保存したりする機能 (この機能を使用できるかどうかは、使用しているプラグインによって決まります)。
- コード印刷機能
- テキスト補完 (コード入力時)

「コード・エディタのキーボード・ショートカット」 578 ページを参照してください。

♦ 別ウィンドウでコード・エディタを開くには、次の手順に従います。

1. Sybase Central の左または右のウィンドウ枠で、ストアド・プロシージャ、ビュー、トリガなどのデータベース・オブジェクトを選択します。
2. [ファイル] – [新しいウィンドウで編集] を選択するか、[Ctrl+E] キーを押します。

コード・エディタのカスタマイズ

[オプション] ダイアログを使用して、コード・エディタの表示をカスタマイズできます。このダイアログを使用して変更できる設定は、フォアグラウンドとバックグラウンドの色と、コード・エディタ全体の外観です。ユーザが加える変更は、次のセッションに移行しても有効です。

♦ **[SQL] タブの編集中にコード・エディタの設定を設定するには、次の手順に従います。**

1. [ファイル] メニューから、[エディタのカスタマイズ] を選択します。
2. 各タブで設定を行います。[OK] をクリックします。

♦ **別のウィンドウで編集中にコード・エディタの設定を設定するには、次の手順に従います。**

1. コード・エディタで [ツール] – [オプション] を選択します。
2. 各タブで設定を行います。[OK] をクリックします。

コード・エディタのキーボード・ショートカット

Sybase Central には、コード・エディタ用に次のキーボード・ショートカットが用意されています。

ファンクション・キー	説明
[Alt + F4]	コード・エディタを閉じる (別のウィンドウで編集中の場合)、または Sybase Central を閉じる (Sybase Central の右ウィンドウ枠でテキストを編集中の場合)
[Backspace]	選択されている内容を削除する。何も選択されていない場合は、このキーを押すとカーソルの左にある 1 文字が削除されます。
[Ctrl +]]	カーソルを閉じカッコまで移動する。カッコ、大カッコ、角カッコ、山カッコに一致するときに、このショートカットを使用します。
[Ctrl + A]	コード・エディタのウィンドウの内容全体を選択する
[Ctrl + Backspace]	カーソルの左の 1 ワードを削除する
[Ctrl + C]	選択したテキストをクリップボードにコピーする
[Ctrl + Del]	カーソルの右の 1 ワードを削除する
[Ctrl + End]	カーソルをコード・エディタのウィンドウの一番下に移動する
[Ctrl + F]	指定したテキストを検索する
[Ctrl + F3]	現在選択されているテキストの次の出現箇所を検索する

ファンクション・キー	説明
[Ctrl + G]	[ジャンプ] ダイアログが開き、ジャンプ先を指定できる
[Ctrl + Home]	カーソルをコード・エディタのウィンドウの先頭に移動する
[Ctrl + L]	現在の行を削除する
[Ctrl + 左矢印]	カーソルを 1 ワード分戻す
[Ctrl + N]	コード・エディタのウィンドウの内容をクリアし、現在のファイルを閉じる (存在する場合)。Sybase Central の右ウィンドウ枠にある [SQL] タブでテキストを編集中であるときは、このショートカットは機能しません。
[Ctrl + E]	コード・エディタが別ウィンドウとして開かれている場合に、ファイルを開く。このショートカットは、Sybase Central の右ウィンドウ枠の [SQL] タブからは使用できません。
[Ctrl + P]	[SQL 文] ウィンドウ枠の内容を印刷する。印刷テキストの外観は、Interactive SQL の [オプション] ダイアログで設定できます。
[Ctrl + 右矢印]	カーソルを 1 ワード分進める
[Ctrl + S]	[SQL 文] ウィンドウ枠の内容を保存する
[Ctrl + Shift +]]	選択範囲を閉じカッコまで拡張する。カッコ、大カッコ、角カッコ、山カッコに一致するときに、このショートカットを使用します。
[Ctrl + Shift + End]	選択範囲をコードの終わりまで拡張する
[Ctrl + Shift + F3]	現在選択されているテキストより前の出現箇所を検索する
[Ctrl + Shift + Home]	選択範囲をコードの先頭まで拡張する
[Ctrl + Shift + L]	現在の行を削除する
[Ctrl + Shift + 左矢印]	選択範囲を 1 ワード分逆方向に拡張する
[Ctrl + Shift + 右矢印]	選択範囲を 1 ワード分先まで拡張する
[Ctrl + Shift + U]	選択している内容を大文字に変換する
[Ctrl + U]	選択している内容を小文字に変換する
[Ctrl + V]	クリップボードの内容を現在のカーソル位置に挿入する
[Ctrl + X]	選択したテキストを切り取る
[Ctrl + Y]	直前に取り消した操作をやり直す

ファンクション・キー	説明
[Ctrl + Z]	最後の操作を取り消す
[Delete]	選択されている内容を削除する
[下矢印]	カーソルを 1 行下に移動する
[End]	カーソルを現在の行の末尾に移動する
[F3]	[検索／置換] ダイアログを開く。現在のウィンドウでテキストを検索していない場合は、指定したテキストを検索して置換できます。それ以外の場合は、指定したテキストの次の出現箇所を検索します。
[Home]	カーソルを現在の行の行頭、または現在の行のテキストの先頭に移動する
[左矢印]	カーソルを 1 文字左へ移動する
[Page Down]	カーソルを現在のページの末尾に移動する
[Page Up]	カーソルを現在のページの先頭に移動する
[右矢印]	カーソルを 1 文字右に移動する
[Shift + 下矢印]	選択範囲を 1 行下まで拡張する
[Shift + End]	現在の行を選択する
[Shift + F3]	[検索／置換] ダイアログを開く。テキストが選択されていない場合は、指定したテキストを検索して置換できます。テキストが選択されている場合は、そのテキストの前の出現箇所を検索します。
[Shift + F10]	フォーカスのある領域のコンテキスト・メニューを表示する。 このキーボード・ショートカットは、領域を右クリックする代わりに使用します。
[Shift + Home]	選択範囲を現在の行のテキストの先頭まで拡張する
[Shift + 左矢印]	選択範囲を、現在選択している文字の左隣の文字まで拡張する
[Shift + Page Down]	選択範囲を 1 ページ分下へ拡張する
[Shift + Page Up]	選択範囲を 1 ページ分上へ拡張する
[Shift + 右矢印]	選択範囲を、現在選択している文字の右隣の文字まで拡張する
[Shift + 上矢印]	カーソルを 1 行分上へ拡張する

ファンクション・キー	説明
[上矢印]	カーソルを 1 行上へ移動する

ログ・ビューワの使用

ログ・ビューワは Sybase Central のダイアログで、製品からのメッセージを表示して格納します。次のようなタイプのメッセージを表示します。

- **情報** 現在のセッションに関する基本的な情報メッセージ
- **警告** 発生したアクションに対する警告メッセージ
- **エラー** 失敗したアクションに対するエラー・メッセージ

表示するメッセージのタイプや番号を制限したり、特定のプラグインからのメッセージだけを表示したりできます。また、メッセージをファイルに保存したり、すべてのメッセージをリストからクリアしたりすることもできます。

Sybase Central で作業している場合は、[ツール] メニューからログ・ビューワにアクセスできます。

- **ログ・ビューワを開くには、次の手順に従います。**

1. Sybase Central で [ツール] – [ログ・ビューワ] を選択します。

   Sybase Central ログ・ビューワが開き、現在メッセージがあればそれを表示します。
2. ログ・ビューワを使用して、ログに記録するメッセージのタイプを設定します。

SQL Anywhere プラグインの使用

SQL Anywhere プラグインを使用して、既存データベースのアップグレード、新しいデータベースの作成、データベースの管理を行うことができます。必要なモードを選択するには、[モード] メニュー、またはモードに対応するツールバーのボタンを使用します。

SQL Anywhere プラグインは、次のいずれかのモードで操作します。

- **設計モード** 設計モードでは、テーブル、ユーザ、トリガ、インデックス、リモート・サーバなどのデータベース・オブジェクトを作成したり、変更したりできます。また、テーブルへのデータの追加、新しいデータベースの作成、既存データベースのアップグレードを行うことも可能です。

  設計モードで操作中に SQL Anywhere データベースで実行可能なタスクの詳細については、「データベース・オブジェクトの使用」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **デバッグ・モード**　デバッグ・モードでは、SQL Anywhere のデバッガを使用して、SQL のストアド・プロシージャ、トリガ、イベント・ハンドラの開発に役立てることができます。

  デバッグ・モードを使用する方法の詳細については、「プロシージャ、関数、トリガ、イベントのデバッグ」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **アプリケーション・プロファイリング・モード**　アプリケーション・プロファイリング・モードでは、データベースのアプリケーション・プロファイリングや診断トレーシングを設定できます。生成されるデータは、アプリケーションがデータベースとやりとりする方法を理解するのに役立ちます。また、パフォーマンスの問題を特定し、解消するためにも有用です。

  アプリケーション・プロファイリング・モードを使用する方法の詳細については、「アプリケーション・プロファイリング」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

参照

- 「Sybase Central、Interactive SQL、SQL Anywhere コンソール・ユーティリティからの接続」 64 ページ
- 「データベース・オブジェクトの使用」『SQL Anywhere サーバ - SQL の使用法』
- 「Sybase Central を使用した Windows CE データベースの作成」 1047 ページ

SQL Anywhere プラグインのデータベース・オブジェクトのコピー

SQL Anywhere プラグインでは、既存のデータベース・オブジェクトをコピーして、同じデータベース内の別のロケーションか、まったく別のデータベースに挿入できます。

オブジェクトをコピーするには、Sybase Central の左ウィンドウ枠でオブジェクトを選択し、該当するフォルダまたはコンテナにドラッグします。または、オブジェクトをコピーして、該当するフォルダまたはコンテナに貼り付けます。新しいオブジェクトが作成され、元のオブジェクトのコードが新しいオブジェクトにコピーされます。同じデータベース内でオブジェクトをコピーする場合は、新しいオブジェクトの名前を変更してください。

データベース内の別のオブジェクトにオブジェクトを貼り付けることもできます。たとえば、テーブルをユーザに貼り付けると、このテーブルに対するパーミッションがこのユーザに与えられます。

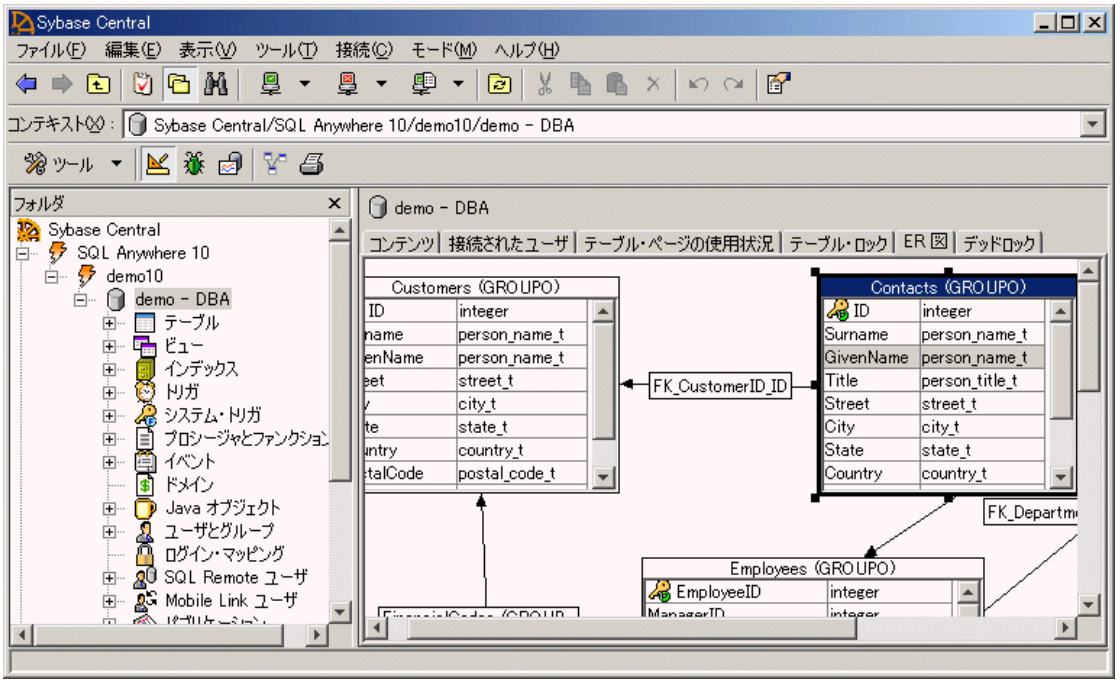
Sybase Central では、次にリストされているオブジェクトのいずれかをコピーすると、そのオブジェクトの SQL がクリップボードにコピーされるため、Interactive SQL やテキスト・エディタなどの他のアプリケーションに貼り付けることができます。たとえば、Sybase Central でインデックスをコピーし、テキスト・エディタに貼り付けると、そのインデックスの CREATE INDEX 文が表示されます。SQL Anywhere プラグインにある次のオブジェクトをコピーできます。

- アーティクル
- 検査制約
- カラム
- DB 領域
- ディレクトリ・アクセス・サーバ
- ドメイン
- イベント
- 外部ログイン

- 外部キー
- インデックス
- ログイン・マッピング (統合化ログインと Kerberos ログイン)
- メンテナンス・プラン
- メンテナンス・プラン・レポート
- メッセージ型
- Mobile Link ユーザ
- プライマリ・キー
- プロシージャとファンクション
- パブリケーション
- リモート・サーバ
- スケジュール
- SQL Remote サブスクリプション
- 同期サブスクリプション
- システム・トリガ
- テーブル
- トリガ
- 一意性制約
- ユーザとグループ
- ビュー
- Web サービス

SQL Anywhere プラグインからの ER 図の表示

SQL Anywhere プラグインからデータベースに接続すると、データベース内のテーブルの ER 図を表示できます。データベースを選択すると、右ウィンドウ枠の [ER 図] タブをクリックして、図を参照できます。



図内のオブジェクトの配置を調整すると、その内容は Sybase Central セッション間で保持されます。テーブルをダブルクリックすると、対象テーブルのカラム定義を参照できます。

図のテーブルは、データベースのフィルタリング設定に従って表示されます。フィルタリングは所有者別に行われます。

♦ ER 図に含まれるテーブルを変更するには、次の手順に従います。

1. Sybase Central の左ウィンドウ枠またはツールバー・ドロップダウン・リストでデータベースを選択し、[ファイル] – [所有者別にオブジェクトをフィルタ] を選択します。

   ER 図で参照したいテーブルのデータベース・ユーザを選択します。

2. 右ウィンドウ枠の [ER 図] タブをクリックします。

   ER 図が表示されます。

3. [ファイル] – [ER 図のテーブルを選択] を選択します。

   [ER 図のテーブルを選択] ダイアログが表示されます。このダイアログを使用して、ER 図に表示されるテーブルをカスタマイズできます。

   [追加] および [削除] ボタンを使用して、[選択したテーブル] リストに表示されるテーブルをカスタマイズします。

参照

- 「ER (実体関連) 図」 『SQL Anywhere サーバ - SQL の使用法』

Interactive SQL

Interactive SQL は SQL Anywhere に同梱されているツールです。このツールを使用すると、SQL Anywhere と Ultra Lite データベースに対して SQL 文の実行、スクリプトのビルド、データベースのデータの表示を実行できます。Interactive SQL は、次の目的で使用できます。

- SQL 文のデータベース・サーバへの送信。「Interactive SQL からの SQL 文の実行」 589 ページを参照してください。
- データベース内の情報のブラウズ。「Interactive SQL からの SQL 文の実行」 589 ページを参照してください。
- 結果セットにあるデータの編集。「Interactive SQL での結果セットの編集」 595 ページを参照してください。
- データベースへのデータの読み込み。「Interactive SQL の [インポート] ウィザードの使用」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- クエリ結果のファイルへのエクスポート。「クエリ結果のエクスポート」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- コマンド・ファイルまたはスクリプト・ファイルの実行。「SQL コマンド・ファイルの実行」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- インデックス・コンサルタント (クエリ・パフォーマンスの向上に役立つツール) の実行。「インデックス・コンサルタント」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- クエリ・エディタ (あらゆる種類のクエリの設計、分析、テストに役立つツール) へのアクセス。「クエリ・エディタのヘルプ」 『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

Interactive SQL は、Windows、Solaris、Linux、Mac OS X で利用できます。

Interactive SQL のみから使用される SQL 文

Interactive SQL は、Interactive SQL でのみ使用可能な SQL 文だけでなく、SQL Anywhere と Ultra Lite データベースでサポートされるすべての SQL 文をサポートします。

- 「CLEAR 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「CONFIGURE 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「CONNECT 文 [ESQL] [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「DESCRIBE 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「DISCONNECT 文 [ESQL] [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「EXIT 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「HELP 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「INPUT 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「OUTPUT 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「PARAMETERS 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「READ 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

- 「SET CONNECTION statement [Interactive SQL] [ESQL]」『SQL Anywhere サーバ - SQL リファレンス』
- 「SET OPTION 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』
- 「START ENGINE 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』
- 「START LOGGING 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』
- 「STOP LOGGING 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』
- 「SYSTEM 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』

Interactive SQL の起動

Interactive SQL は、コマンド・プロンプト、Windows の [スタート] メニュー、Sybase Central から起動できます。

♦ Interactive SQL を起動するには、次の手順に従います (コマンド・プロンプトの場合)。

· コマンド・プロンプトから dbisql コマンドを実行します。

データベースへの接続パラメータを指定する -c オプションを省略するなど、接続パラメータの指定が不十分であると、[接続] ダイアログが表示されます。そのダイアログに、データベースに対する接続情報を入力できます。

サポートされるオプションの詳細については、「Interactive SQL ユーティリティ (dbisql)」 672 ページを参照してください。

次のコマンドは、Interactive SQL を起動し、サンプル・データベースに接続します。

```
dbisql -c "UID=DBA;PWD=sql;DSN=SQL Anywhere 10 Demo"
```

♦ Interactive SQL を起動するには、次の手順に従います (Windows の場合)。

1. [スタート] メニューから、[プログラム] − [SQL Anywhere 10] − [Interactive SQL] の順に選択します。

   Interactive SQL が開き、[接続] ダイアログが表示されます。

2. [接続] ダイアログで、データベースの接続情報を入力します。

♦ Interactive SQL を起動するには、次の手順に従います (Sybase Central の場合)。

1. [ツール] メニューから、[SQL Anywhere 10] − [Interactive SQL を開く] の順に選択します。

   Interactive SQL が開き、[接続] ダイアログが表示されます。

2. [接続] ダイアログで、データベースの接続情報を入力します。

Linux のデスクトップ・アイコンをサポートするバージョンの Linux を使用している場合で、SQL Anywhere 10 をインストールするときにこれらのアイコンをインストールするように選択したときは、次の手順を使用できます。

♦ **Interactive SQL を起動するには、次の手順に従います (Linux デスクトップ・アイコンの場合)。**

1. デスクトップで [SQL Anywhere 10] フォルダを開きます。
2. Interactive SQL をダブルクリックします。

   Interactive SQL が開き、[接続] ダイアログが表示されます。
3. [接続] ダイアログで、データベースの接続情報を入力します。

注意
以降の手順は、SQL Anywhere ユーティリティのソースを指定済みであることを前提としています。「UNIX と Mac OS X での環境変数の設定」 306 ページを参照してください。

♦ **Interactive SQL を起動するには、次の手順に従います (UNIX コマンド・ラインの場合)。**

1. ターミナル・セッションで次のコマンドを入力します。

   ```
   dbisql
   ```

   Interactive SQL が開き、[接続] ダイアログが表示されます。
2. [接続] ダイアログで、データベースの接続情報を入力します。

ヒント
以下の手順で Sybase Central から Interactive SQL にアクセスすることもできます。

- データベースを選択し、[ファイル] メニューから [Interactive SQL を開く] を選択する。
- データベースを右クリックし、ポップアップ・メニューから [Interactive SQL を開く] を選択する。
- ストアド・プロシージャを右クリックし、ポップアップ・メニューから [Interactive SQL から実行] を選択する。Interactive SQL は [SQL 文] ウィンドウ枠にプロシージャへの呼び出しを表示し、ストアド・プロシージャを実行します。
- テーブルを右クリックし、ポップアップ・メニューで [Interactive SQL によるデータ表示] を選択する。SELECT * FROM *table-name* を表示した Interactive SQL が開かれ、クエリが実行されます。

Interactive SQL メイン・ウィンドウの説明

Interactive SQL のウィンドウは、2 つのウィンドウ枠に分かれています。

♦ **SQL 文**　このウィンドウ枠には、データのアクセスや変更に使用する SQL 文を入力します。

- **結果**　[結果] ウィンドウ枠には、[結果]、[メッセージ]、[プラン] の 3 つのタブがあります。これらのタブは、[結果] ウィンドウ枠の下部に表示されます。

  [結果] タブには、実行したコマンドの結果が表示されます。たとえば、データベースから特定のデータを検索する SQL 文を使用すると、その上のウィンドウ枠の [結果] タブに、検索条件と一致するカラムとローが表示されます。[結果] タブに表示された結果セットは編集できます。「Interactive SQL での結果セットの編集」 595 ページを参照してください。

  [メッセージ] タブには、Interactive SQL で実行する SQL 文についてのデータベース・サーバのメッセージが表示されます。

  [プラン] タブには、SQL 文についてのクエリ・オプティマイザの実行プランが表示されます。「グラフィカルなプラン」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

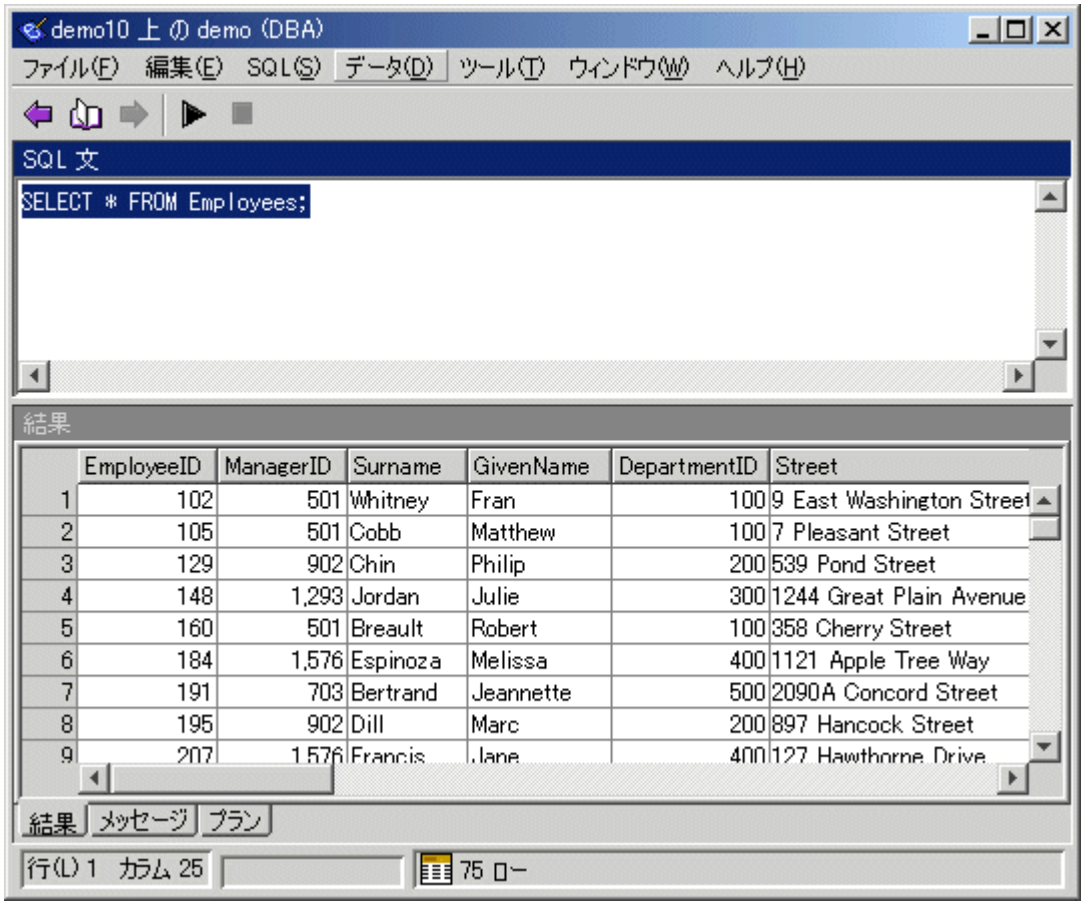
Interactive SQL では、[オプション] ダイアログを使用してタブやウィンドウ枠の設定を変更できます。このダイアログには、[ツール] – [オプション] を選択してアクセスします。

Interactive SQL からデータベースに接続しているとき、ウィンドウ上部にあるタイトル・バーには次のような接続情報が表示されます。

server-name 上の *database-name* (*userid*)

たとえば、SQL Anywhere 10 Sample ODBC を使用してサンプル・データベースに接続すると、タイトル・バーには次のような情報が含まれます。

demo10 上の demo (DBA)



Interactive SQL からの SQL 文の実行

Interactive SQL の重要な使用目的の 1 つは、テーブル・データをブラウズすることです。Interactive SQL では、データベース・サーバに要求を送信することによって情報を検索します。一方、データベースサーバは、情報を調べ、それを Interactive SQL に返します。

SELECT 文を実行すると、[結果] ウィンドウ枠の [結果] タブに結果セットが表示されます。デフォルトでは、ロー番号が結果セットの左に表示されます。

♦ **SQL 文を実行するには、次の手順に従います。**

1. [SQL 文] ウィンドウ枠にクエリを入力します。
2. [SQL 文の実行] をクリックする、[SQL] – [実行] を選択する、[F5] キーを押す、のいずれかの操作を行って、文を実行します。

♦ **[SQL 文] ウィンドウ枠をクリアするには、次の手順に従います。**

- [編集] メニューから [SQL のクリア] を選択するか、[Ecs] キーを押します。

複数の文の実行

各文がコマンド・デリミタで終了している間は、Interactive SQL から複数の SQL 文を実行できます。コマンド・デリミタは command_delimiter オプションで設定されるもので、デフォルトではセミコロン (;) です。「command_delimiter オプション [Interactive SQL]」 611 ページを参照してください。

セミコロンを使用する代わりに、セパレータ **go** を独立した行の先頭に配置することもできます。

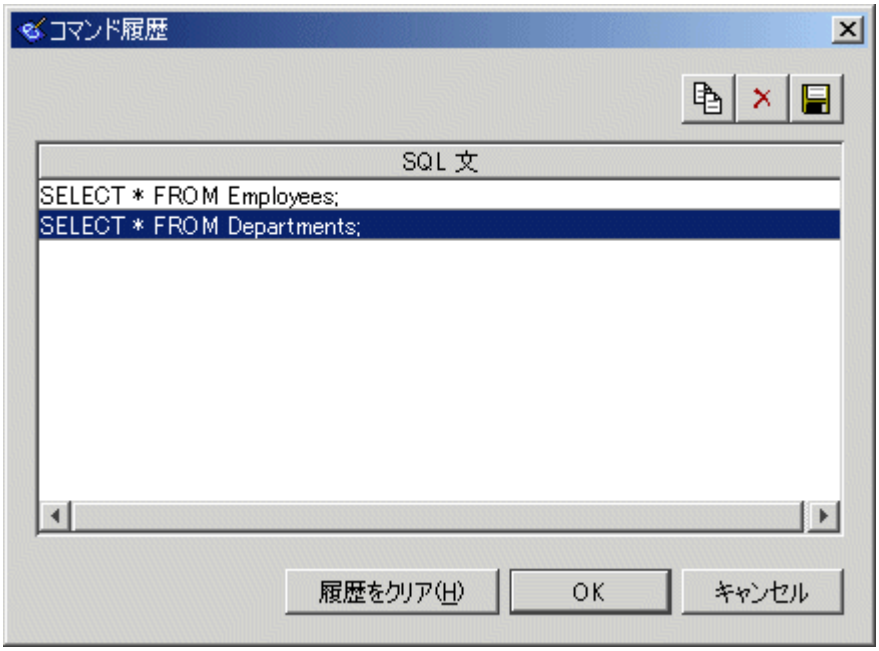
デフォルトでは、Interactive SQL は、最後に実行された文の最初の結果セットを表示します。最後の文のすべての結果セットを参照するには、[オプション] ダイアログの [結果] タブで [複数の結果セットを表示] オプションを選択します。

ヒント
[F9] キーを押すと、[SQL 文] ウィンドウ枠で選択したテキストだけを実行できます。

コマンドの再呼び出し

コマンドを実行すると、Interactive SQL は自動的にそのコマンドを履歴リストに保存し、次の Interactive SQL セッションまで保持します。Interactive SQL は、最近使用したコマンドを 50 個まで記録します。

[コマンド履歴] ダイアログで、コマンド・リスト全体を表示できます。[コマンド履歴] ダイアログにアクセスするには、[Ctrl + H] キーを押すか、ツールバーのブック・アイコンをクリックします。



最後に使用したコマンドはリストの一番下に表示されます。コマンドを再度呼び出すには、そのコマンドを選択して [OK] をクリックします。コマンドが Interactive SQL の [SQL 文] ウィンドウ枠に表示されます。[コマンド履歴] ダイアログから複数のコマンドを選択できます。

また、[コマンド履歴] ダイアログを使用しないでコマンドを再度呼び出すこともできます。ツールバーの矢印を使用してコマンドを前後にスクロールするか、または [Alt + 右矢印] キーと [Alt + 左矢印] キーを押します。

注意
パスワード情報を含む SQL 文 (GRANT CONNECT、GRANT REMOTE DBA、CONNECT、または CREATE EXTERNLOGIN) を実行する場合は、現在の Interactive SQL セッションの間、[コマンド履歴] ダイアログにパスワード情報が表示されます。
コマンド履歴がその後の Interactive SQL セッションで表示された場合は、パスワード情報を含むこれらの文でパスワードが「...」に置換されます。たとえば、Interactive SQL で次の文を実行したと仮定します。

```
GRANT CONNECT TO testuser
 IDENTIFIED BY testpassword
```

この場合は、後続の Interactive SQL セッションで [コマンド履歴] ダイアログに次のように表示されます。

```
GRANT CONNECT TO testuser
 IDENTIFIED BY ...
```


[コマンド履歴] ダイアログからのコマンドのコピー

[コマンド履歴] ダイアログからコマンドをコピーして、他の場所で使用できます。複数のコマンドをコピーする場合は、それらのコマンドがコマンド・デリミタ (デフォルトではセミコロン) で区切られます。

♦ **[コマンド履歴] ダイアログからコマンドをコピーするには、次の手順に従います。**

1. [コマンド履歴] ダイアログを開きます。
2. 1 つまたは複数のコマンドを選択し、[Ctrl+C] キーを押すか、[コピー] をクリックします。
3. [OK] をクリックして、選択した文を Interactive SQL の [SQL 文] ウィンドウ枠にコピーします。

[コマンド履歴] ダイアログからのコマンドの保存

コマンドは、以降 Interactive SQL セッションで使用できるように、テキスト・ファイルに保存することも可能です。

♦ **コマンド履歴をファイルに保存するには、次の手順に従います。**

1. [コマンド履歴] ダイアログを開きます。
2. [保存] ボタンをクリックするか、[Ctrl + S] キーを押します。
3. [名前を付けて保存] ダイアログで、ファイルの保存場所と名前を指定します。

   コマンド履歴ファイルの拡張子は、*.sql* にします。
4. 設定を終了したら、[保存] をクリックします。

[コマンド履歴] ダイアログからのコマンドの削除

[コマンド履歴] ダイアログの内容は、Interactive SQL セッション間で保持されます。コマンドは、次の 2 つの方法のいずれかで履歴から削除できます。

- 1 つまたは複数のコマンドを選択し、[削除] ボタンをクリックするか [Delete] キーを押して、選択したコマンドをダイアログから削除します。この操作は取り消せません。
- [履歴をクリア] をクリックすると、すべてのコマンドがダイアログから削除されます。この操作は取り消せません。

コマンドのロギング

Interactive SQL のロギング機能を使うとコマンドの実行を記録できます。Interactive SQL は、ロギング・プロセスを停止するまで、または現在のセッションを終了するまで、記録を続けます。記録されたコマンドは、コマンドを再び使用できるように、ログ・ファイルに保存されます。

♦ **Interactive SQL コマンドのロギングを開始するには、次の手順に従います。**

1. [SQL] メニューから、[ロギングの開始] を選択します。

2. [名前を付けて保存] ダイアログで、ログ・ファイルの保存場所と名前を指定します。たとえば、*filename.sql* のようなファイル名を指定します。
3. 設定を終了したら、[保存] をクリックします。

♦ **Interactive SQL コマンドのロギングを停止するには、次の手順に従います。**

· [SQL] メニューから、[ロギングの停止] を選択します。

ヒント
ロギングの開始と停止には、[SQL 文] ウィンドウ枠にコマンドを入力する方法もあります。ロギングを開始するには、「**START LOGGING** *'c:¥filename.sql'*」と入力して実行します。*c:¥filename.sql* には、ログ・ファイルのパス、名前、拡張子を指定します。一重引用符が必要なのはパスにスペースが含まれる場合だけです。ロギングを停止するには、「**STOP LOGGING**」と入力して実行します。
ロギングを開始すると、正常に動作しなかったものも含めて実行したコマンドがすべて記録されます。

Interactive SQL のコマンドのキャンセル

キャンセル操作によって現在の処理が停止し、次のコマンドのプロンプトが表示されます。Interactive SQL ツールバーで [中断] ボタンをクリックすると、コマンドをキャンセルできます。

コマンド・ファイルが処理中の場合、または [SQL 文] ウィンドウ枠に複数の文がある場合は、必要なアクション (コマンド・ファイルを停止するか、継続するか、Interactive SQL を終了する) を指定するためのプロンプトが表示されます。これらのアクションは、Interactive SQL の on_error オプションを使用して制御できます。「on_error オプション [Interactive SQL]」 620 ページを参照してください。

コマンド・ファイルでの Interactive SQL の使用

コマンド・ファイルは、SQL 文を含むテキスト・ファイルであり、同じ SQL 文を繰り返し実行する場合に便利です。Interactive SQL を使用して、コマンド・ファイルを開いたり、表示、実行、エクスポートを実行したりできます。

コマンド・ファイルは、次のいずれかの方法で Interactive SQL から実行できます。

- Interactive SQL の READ 文を使用する
- [ファイル] – [スクリプトの実行] を選択する
- コマンド・ファイルを、Interactive SQL のコマンド・ライン引数として指定する

Windows プラットフォームでは、Interactive SQL を *.sql* コマンド・ファイルのデフォルト・エディタにできます。このように設定すると、ファイルをダブルクリックするだけで、Interactive SQL の [SQL 文] ウィンドウ枠に内容が表示されます。

♦ **Interactive SQL を .sql ファイルのデフォルト・エディタにするには、次の手順に従います。**

1. [Interactive SQL] から、[ツール] – [オプション] を選択します。

   [オプション] ダイアログが表示されます。

2. [一般] タブで、[Interactive SQL を .SQL ファイルとプラン・ファイルのデフォルト・エディタにする] オプションを選択します。

3. [OK] をクリックします。

コマンド・ファイルで Interactive SQL を使用する方法の詳細については、次の項を参照してください。

- 「SQL コマンド・ファイルの使用」『SQL Anywhere サーバ - SQL の使用法』
- 「SQL コマンド・ファイルの実行」『SQL Anywhere サーバ - SQL の使用法』
- 「SQL コマンド・ファイルのロード」『SQL Anywhere サーバ - SQL の使用法』
- 「ファイルへのデータベース出力の書き込み」『SQL Anywhere サーバ - SQL の使用法』

テーブル、カラム、プロシージャの検索

Interactive SQL にコマンドを入力する際現在のデータベースに格納されているテーブル名、カラム名、またはプロシージャ名を検索し、それをカーソル位置に挿入することができます。

♦ **データベース内のテーブル名を検索するには、次の手順に従います。**

1. [ツール] メニューから、[テーブル名のルックアップ] を選択するか、[F7] キーを押します。
2. テーブルを検索し、選択します。
3. [OK] をクリックして、テーブル名を [SQL 文] ウィンドウ枠の現在のカーソル位置に挿入します。

♦ **データベース内のカラム名を検索するには、次の手順に従います。**

1. [ツール] メニューから、[テーブル名のルックアップ] を選択するか、[F7] キーを押します。
2. カラムを含むテーブルを検索し選択します。
3. [カラムを表示] をクリックします。
4. カラムを選択し [OK] をクリックして、カラム名を [SQL 文] ウィンドウ枠の現在のカーソル位置に挿入します。

♦ **データベース内のプロシージャ名を検索するには、次の手順に従います。**

1. [ツール] メニューから、[プロシージャ名のルックアップ] を選択するか、[F8] キーを押します。
2. プロシージャを検索し、選択します。

3. [OK] をクリックして、プロシージャ名を [SQL 文] ウィンドウ枠の現在のカーソル位置に挿入します。

[テーブル名のルックアップ] と [プロシージャ名のルックアップ] の各ダイアログで、検索するテーブルまたはプロシージャの最初の数文字を入力します。入力した文字で始まる項目だけを含むようにリストが限定されます。

SQL のワイルドカード文字 '%' (パーセント記号) と '_' (アンダースコア) を使用すると、検索対象を絞り込むことができます。'%' は、0 文字以上の任意の文字列を表し、'_' は、任意の 1 文字を表します。

たとえば、profile という語を含むすべてのテーブルをリストするには、「**%profile%**」と入力します。

テーブル名に含まれるパーセント記号またはアンダースコアを検索する場合は、パーセント記号またはアンダースコアの前にエスケープ文字を付ける必要があります。iAnywhere JDBC ドライバを使用している場合は '~' (チルダ) をエスケープ文字として使用します。

ヒント
[SQL 文] ウィンドウ枠に入力するとき、Interactive SQL によってデータベース・オブジェクト名のテキスト補完がサポートされます。この機能は、テーブルやプロシージャの名前を検索するときの代替手段として使用できます。「テキスト補完の使用」 625 ページを参照してください。

SQL 文の印刷

[SQL 文] ウィンドウ枠の内容を印刷するには、[Ctrl + P] キーを押すか、[ファイル] メニューから [印刷] を選択します。Interactive SQL の [オプション] ダイアログ ([エディタ] ページで [印刷] タブをクリック) で、ヘッダまたはフッタの追加、その他のフォーマット・オプションの設定ができます。「[印刷] タブ」 『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

Interactive SQL でのグラフィカル・プランの印刷については、「グラフィカルなプラン」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

Interactive SQL での結果セットの編集

Interactive SQL でクエリを実行すると、結果セットを編集してデータベースを修正できます。また、結果セットからローを選択し、他のアプリケーションで使用できるようにコピーすることもできます。Interactive SQL は、ローの編集、挿入、削除をサポートしています。結果セットの編集には、UPDATE、INSERT、DELETE の各文を実行した場合と同様の効果があります。

結果セット内のローまたは値を編集するには、値を修正するテーブルまたはカラムに対する適切なパーミッションが必要です。たとえば、ローを削除する場合は、そのローが属しているテーブルに対する DELETE パーミッションが必要です。

次の場合には、結果セットを編集できません。

- プライマリ・キーを持つテーブルからカラムを選択したが、一部のプライマリ・キー・カラムを選択していない
- JOIN の結果セットを編集しようとした (たとえば、結果セットに複数のテーブルのデータがある場合)

次の場合には、結果セットの編集に失敗することがあります。

- パーミッションのないローやカラムを編集しようとした
- 無効な値を入力した (たとえば、数値カラムに文字列を入力したり、NULL を使用できないカラムに NULL を入力した場合)

編集に失敗すると、エラーを説明する Interactive SQL エラー・メッセージが表示されます。データベース・テーブルの値は変更されません。

Interactive SQL の結果セットからテーブル値を編集する

Interactive SQL から、データベース・テーブルに存在するローの一部またはすべての値を変更できます。修正するカラムに対する UPDATE パーミッションが必要です。結果セットを編集する場合、一度に変更できるのは 1 つのローの値だけです。

♦ 結果セット内のローを編集するには、次の手順に従います。

1. Interactive SQL でクエリを実行します。
2. [結果] タブで、変更したい値をクリックします。
3. 値を右クリックし、ポップアップ・メニューから [編集] を選択するか、[F2] キーを押して、結果セットを編集します。

   その値を含むテーブル・セルに点滅するカーソルが表示されます。
4. 新しい値を入力します。そのローの他の値を変更する場合は、[Tab] または [Shift + Tab] キーを押して他の値に移動します。
5. ローの値を編集したら、[Enter] キーを押してデータベースを更新します。

   [Esc] キーを押すと、選択した値に対して行った変更をキャンセルできます。
6. COMMIT 文を実行し、テーブルに対する変更を永続的なものにします。

Interactive SQL の結果セットからデータベースにローを挿入する

Interactive SQL では、テーブルに新しいローを追加できます。結果セット内のカラム間をタブで移動し、ローに新しい値を追加します。新しいローを追加するには、テーブルに対する INSERT パーミッションが必要です。

♦ **結果セットに新しいローを挿入するには、次の手順に従います。**

1. 結果セットを右クリックし、ポップアップ・メニューで [追加] を選択します。

   新しい空白のローが表示され、そのローの最初の値に点滅するカーソルが表示されます。

2. 新しい値を入力します。

   カラムには、無効なデータ型は入力できません。たとえば、INT データ型を受け入れるカラムには、文字列は入力できません。

3. [Tab] キーを押して次のカラムに進みます。

4. すべてのカラム値が追加されるまで手順 2 と 3 を繰り返します。

5. [Enter] キーを押してデータベースを更新します。

デフォルト値を持つカラムへの値の挿入

デフォルト値を持つカラムに値を追加するとき、セルのエディタには (デフォルト) 項目を含むリストが表示されます。デフォルト値を挿入する場合は、[(デフォルト)] を選択します。同様に、NULL 値を受け入れるカラムである場合、リストには [(NULL)] が表示されます。NULL 値を受け入れず、デフォルト値も持たないカラムである場合は、値を入力する必要があります。

計算カラムへの値の挿入

結果セット内に計算カラムがあり、計算カラムに値を指定しない場合、値はデータベースが更新されたときに計算されます。しかし、計算カラムに値を指定した場合は、指定した値でデータベースが更新され、値は計算カラムに対して計算されません。

INPUT 文を使用した新しいローの挿入

Interactive SQL の結果セットから新しいローを挿入するには、INPUT 文で PROMPT 句を使用してローを追加するという方法もあります。PROMPT 句を指定すると、Interactive SQL によって、テーブルの各カラムの値を入力するよう要求されます。たとえば、Products テーブルに新しいローを追加し、各カラムの値を入力するよう求めるメッセージを表示するには、Interactive SQL で次の文を実行します。

```
INPUT INTO Products PROMPT
```

Interactive SQL を使用してデータベースからローを削除する

また、Interactive SQL では、データベース・テーブルからローを削除できます。ローを削除するには、テーブルに対する DELETE パーミッションが必要です。

♦ **結果セットからローを削除するには、次の手順に従います。**

1. 削除するローを選択します。ローを選択するには、次の手順に従います。

   ♦ [Shift] キーを押しながらローをクリックします。

- [Shift + 上矢印] または [Shift + 下矢印] キーを押します。

ローが連続していない場合は、それぞれ個別に削除します。

2. [Delete] キーを押します。

選択したローがデータベース・テーブルから削除されます。

3. COMMIT 文を実行し、変更を永続的なものにします。

Interactive SQL の結果セットからローをコピーする

Interactive SQL に表示される結果セットからローを直接コピーし、他のアプリケーションに貼り付けることができます。ローをコピーすると、カラム見出しとテーブル・データもクリップボードにコピーされます。コピーしたデータはカンマで区切られ、すべての文字列は一重引用符で囲まれます。

- **Interactive SQL の結果セットからローをコピーするには、次の手順に従います。**

1. コピーするローを選択します。ローを選択するには、次の手順に従います。

   - [Shift] キーを押しながらローをクリックします。
   - [Shift + 上矢印] または [Shift + 下矢印] キーを押します。

2. [Ctrl+C] キーを押して、選択したローをコピーします。

   選択したローが、カラム見出しを含めてクリップボードにコピーされます。それを、他のアプリケーションに貼り付けることもできます。

結果セットからの個々の値のコピー

結果セットから単一の値をコピーするには、値を選択し、結果セットを右クリックし、ポップアップ・メニューで [セルのコピー] を選択します。この操作を行った場合、カラム見出しはコピーされず、データのみがコピーされます。引用もされません。

複数のウィンドウを開く

複数の Interactive SQL ウィンドウを開くことができます。各ウィンドウは、接続するデータベース別になっています。異なるサーバ上にある 2 つ (またはそれ以上) のデータベースに同時に接続したり、単一のデータベースへの同時接続を開始したりできます。

- **新しい Interactive SQL ウィンドウを開くには、次の手順に従います。**

1. [ウィンドウ] メニューから、[新しいウィンドウ] を選択します。

   [接続] ダイアログが表示されます。

ヒント
SQLCONNECT 環境変数が設定されている場合や、すでに SQL Anywhere データベースに接続している場合は、情報の入力を求める前に、サーバがこの情報を使ってデータベースに接続しようとします。同様に、ULCONNECT 環境変数が設定されている場合や、すでに Ultra Lite データベースに接続している場合は、情報の入力を求める前に、サーバがこの情報を使ってデータベースに接続しようとします。それに失敗した場合、またはデータベースにまだ接続していない場合に、[接続] ダイアログが表示されます。

2. [接続] ダイアログに接続オプションを入力し、[OK] をクリックして接続します。

   接続情報 (データベース名、ユーザ ID、データベース・サーバ名など) が Interactive SQL のタイトル・バーに表示されます。

データベースとの接続を開始または解除するには、[SQL] メニューの [接続] と [切断] を使用する方法や、CONNECT 文または DISCONNECT 文を実行する方法があります。

ソース制御の統合

Interactive SQL はサードパーティのソース制御システムと統合でき、Interactive SQL 内からファイルに対する一般的なソース制御操作を実行できます。Windows の場合、Interactive SQL は、Microsoft の Visual SourceSafe などの Microsoft Common Source Code Control API (SCC) をサポートする、ほとんどのソース制御製品と統合できます。Windows およびその他のプラットフォームでは、Interactive SQL を設定して、SCC API をサポートしないソース制御製品を使用することもできます。その場合、ソース制御の各アクションを実行するためのコマンド・ラインを Interactive SQL に提供します。コマンドの出力は、ログ・ウィンドウに表示されます。

Interactive SQL は、次のタスクをサポートします (対象のタスクがソース制御製品でサポートされている場合)。

- ソース制御プロジェクトを開く
- 取得
- チェック・イン
- チェック・アウト
- チェック・アウトの取り消し
- バージョンの比較
- ファイルの履歴の表示
- ファイルのプロパティの表示
- ソース制御マネージャの実行

基本となるソース制御プログラムがアクションをサポートしていない場合は、対応するメニュー項目が無効になります。たとえば、Visual SourceSafe は上記のすべてのアクションをサポートしていますが、カスタム (コマンド・ライン) ソース制御システムを使用すると、ソース制御プロジェクトを開いたり、ソース制御マネージャを実行したりすることはサポートされません。

サポートされるアクションの詳細については、次の項を参照してください。

- 「Interactive SQL からソース制御プロジェクトを開く」 601 ページ

- 「Interactive SQL からファイルをチェック・アウトする」 602 ページ
- 「Interactive SQL からファイルをチェック・インする」 603 ページ
- 「その他のソース制御アクション」 604 ページ

Interactive SQL からソース制御プログラムを使用する前に、その操作を理解しておくことが必要です。

Interactive SQL を設定してソース制御を使用する

ファイルのチェック・インやチェック・アウト、異なるバージョンのファイルの比較、ファイルの履歴の表示など、ファイルに対するソース制御アクションを実行するには、事前に、ソース制御を使用するように Interactive SQL を設定する必要があります。

Microsoft SCC API をサポートするソース制御製品を備えた Windows コンピュータ上で Interactive SQL を実行している場合は、その製品またはカスタム (コマンド・ライン指向の) システムを使用できます。

SCC ソース制御システムの設定

♦ SCC を備えた Windows で Interactive SQL ソース制御を設定するには、次の手順に従います。

1. [ツール] メニューから [オプション] を選択します。

   [オプション] ダイアログが表示されます。
2. [オプション] ダイアログの左ウィンドウ枠で [ソース制御] をクリックします。
3. [ソース制御の統合を有効にする] を選択し、ソース制御拡張機能を有効にします。
4. [OK] をクリックします。

その他のソース制御システムの設定

♦ コマンド・ライン・インタフェースを備えた Interactive SQL ソース制御システムを設定するには、次の手順に従います。

1. [ツール] メニューから [オプション] を選択します。

   [オプション] ダイアログが表示されます。
2. [オプション] ダイアログの左ウィンドウ枠で [ソース制御] をクリックします。
3. [ソース制御の統合を有効にする] を選択し、ソース制御拡張機能を有効にします。
4. [カスタム・ソース制御システム] を選択します。
5. [設定] をクリックします。

   [カスタム・ソース制御オプション] ダイアログが表示されます。このダイアログでは、Interactive SQL に対して、リストされているソース制御アクションを実行するためのコマンド・セットを提供できます。

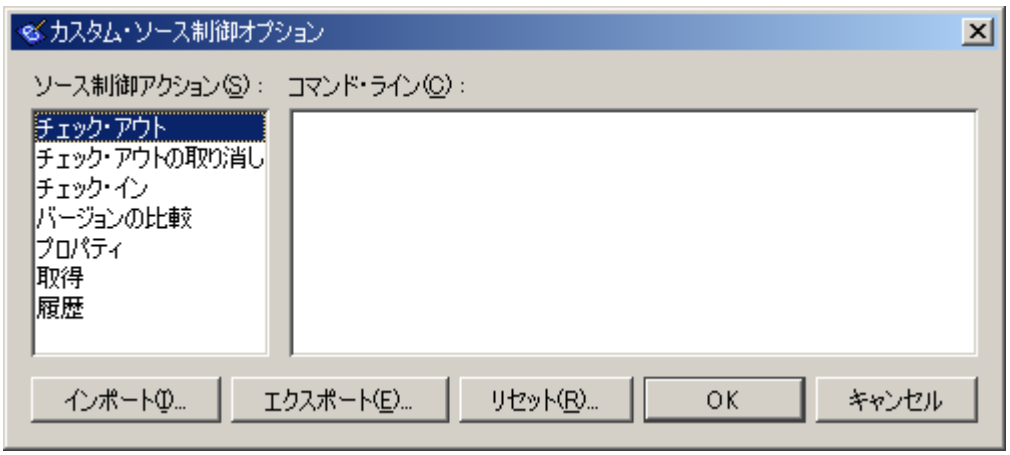
6. [リセット] をクリックします。

   [ソース制御コマンドのリセット] ダイアログが表示されます。

7. リストからソース制御システムを選択し、[OK] をクリックします。

8. [ソース制御アクション] リストからアクションを選択し、必要に応じてリスト内のコマンドを編集します。そして、[コマンド・ライン] ウィンドウ枠で対応するコマンドを入力します。

   [ソース制御アクション] リストに、システム用のコマンドを定義する場合は、プレースホルダ [FILENAME] を使用して、コマンドの実行時に使用するファイル名を表します。たとえば、Perforce でファイルを送信するためのコマンドは p4 submit [FILENAME] です。このコマンドで使用できるアクションは、リストでは太字で示されます。標準のフォントで表示されるアクションは、対象のコマンド用に定義されていません。



   アクションにコマンド・ラインを指定しないと、[ファイル] – [ソース制御] メニューの項目は無効になります。

> **ヒント**
> [カスタム・ソース制御オプション] ダイアログの [エクスポート] をクリックすると、ソース制御コマンド・ラインを外部ファイルにエクスポートできます。このダイアログには、[ツール] – [オプション] を選択し、[ソース制御] ウィンドウ枠で [設定] をクリックしてアクセスできます。[カスタム・ソース制御オプション] ダイアログで [インポート] をクリックすると、後でこれらのコマンドを読むことができます。この機能は、複数のコンピュータ上で Interactive SQL ソース制御コマンド・ラインを設定する必要がある場合に便利です。

9. [OK] をクリックし、もう一度 [OK] をクリックします。

Interactive SQL からソース制御プロジェクトを開く

ソース制御製品の中には、ソース制御アクションを実行する前にソース制御プロジェクトを開くように求められるものがあります。どのプロジェクトを開くことが求められるかという正確な定義は、使用しているソース制御システムによって異なります。一般には、ソース制御下にあるファイルのセットです。ローカルなファイル・システム上に、ファイルの作業用コピーが配置されています。プロジェクトを開くには、通常、ソース制御システムに対するユーザ ID とパスワードのようなクレデンシャルを提供することが求められます。

ソース制御プロジェクトを開くことをサポートするソース制御システムである場合は、[ファイル] – [ソース制御] – [ソース制御プロジェクトを開く] メニュー項目が有効になります。[ファイル] メニューからこのオプションを選択すると、プロジェクトを開くための、ソース制御に固有のダイアログが開きます。一度プロジェクトを開くと、以降の Interactive SQL セッションであっても、同じプロジェクトを再び開く必要はありません。プロジェクトは自動的に開きます。

Interactive SQL からファイルをチェック・アウトする

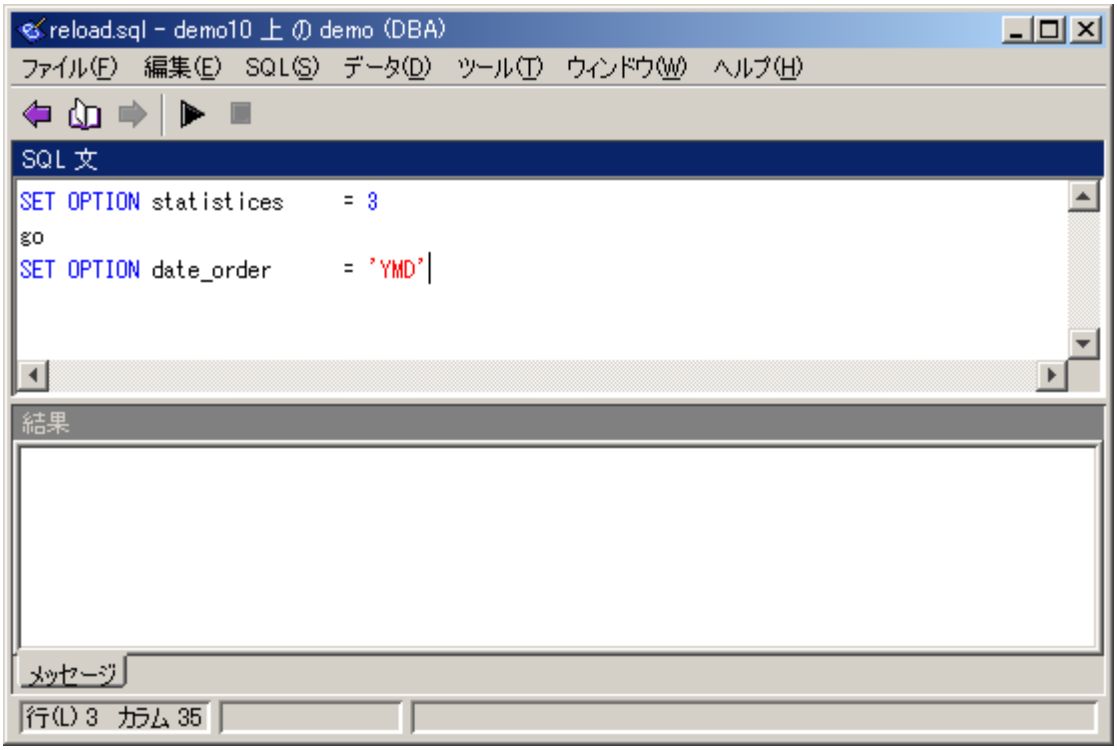
Interactive SQL でファイルを開いた場合、ファイルをチェック・アウトするには 2 つの方法があります。[SQL 文] ウィンドウ枠で内容を変更する方法と、[ファイル] メニューのコマンドを使用する方法です。

Interactive SQL のソース制御オプションを構成するときに [エディタの内容が変更されたら自動的にファイルをチェック・アウト] を選択した場合は、[SQL 文] ウィンドウ枠で内容を変更すると、Interactive SQL によってファイルのチェック・アウトが試行されます。

♦ Interactive SQL の [ファイル] メニューを使用してファイルをチェック・アウトするには、次の手順に従います。

1. [ファイル] – [開く] を選択し、対象のファイルを参照します。

   Interactive SQL ウィンドウ下部にあるステータス・バーに、ファイルのステータスが表示されます。ステータスは、[チェック・イン]、[チェック・アウト]、[Not Controlled] のいずれかになります。チェック・インされているファイルは読み込み専用と見なされ、Interactive SQL のタイトル・バーに [読み込み専用] と表示されます。次の例に、チェック・インされているファイルを示します。



2. [ファイル] – [ソース制御] – [チェック・アウト] を選択して、ファイルをチェック・アウトします。

   使用しているソース制御製品によっては、チェック・アウト手順でコメントの入力やその他のオプション操作を求められることがあります。

警告
SCC に準拠するソース制御システムを使用している場合、必ず正しいステータスが表示されます。ただし、カスタム・ソース制御システムを使用している場合は、ファイルが読み込み専用であるかどうかによってステータスが表示されます。読み込み専用のファイルはチェック・インされていると見なされますが、編集可能なファイルについてはこの前提条件は適用されません。これは、編集可能なファイルはチェック・アウト可能であるか、制御不能であるためです。

Interactive SQL からファイルをチェック・インする

ファイルの編集が完了したら、Interactive SQL からチェック・インできます。

♦ **Interactive SQL からファイルをチェック・インするには、次の手順に従います。**

1. [ファイル] – [ソース制御] – [チェック・イン] を選択します。
2. プロンプトが表示された場合は、チェック・インについてのコメントを入力します。

その他のソース制御アクション

ソース制御プロジェクトを開いたり、ファイルのチェック・インやチェック・アウトを行ったりすることに加えて、Interactive SQL はその他のソース制御アクションをサポートします。これらのアクションの可用性は、使用しているソース制御システムによって異なります。アクションには、Interactive SQL の [ファイル] – [ソース制御] メニューからアクセスします。

- **取得**　このアクションは、[SQL 文] ウィンドウ枠で現在開いているファイルの最後のコピーを取得します。
- **チェック・アウトの取り消し**　ファイルをチェック・アウトした場合で、変更内容を取り消したいときは、[ファイル] – [ソース制御] – [チェック・アウトの取り消し] を選択します。この操作によって、ファイルの作業用コピーは破棄され、ソース制御のアーカイブにあるファイルのコピーがダウンロードされます。
- **バージョンの比較**　このアクションは、開いているファイルの作業用コピーと、ソース制御のアーカイブにあるファイルを比較します。
- **履歴**　このアクションは、開いているファイルに対して実行されたソース制御アクション (通常はチェック・イン) のリストを表示します。
- **プロパティ**　このアクションは、開いているファイルに関連のあるソース制御プロパティのリストを表示します。
- **ソース制御マネージャの実行**　このアクションは、ソース制御システムの管理プログラムを起動します。たとえば、Microsoft Visual SourceSafe を使用している場合、Visual SourceSafe エクスプローラを起動します。

Interactive SQL キーボード・ショートカット

Interactive SQL には、以下のキーボード・ショートカットがあります。

キー	説明
[Alt + F4]	Interactive SQL を終了する
[Alt + 左矢印]	履歴リストにある、前の SQL 文を表示する
[Alt + 右矢印]	履歴リストにある、次の SQL 文を表示する
[Ctrl + Break]	実行中の SQL 文を中断する
[Ctrl + A]	アクティブなウィンドウ枠のすべてのテキストを選択する
[Ctrl + C]	[結果] ウィンドウ枠では、選択したローとカラム見出しをクリップボードにコピーする。 [SQL 文] ウィンドウ枠では、選択したテキストをクリップボードにコピーする。
[Ctrl + End]	現在のウィンドウ枠の一番下に移動する

キー	説明
[Ctrl + F]	[検索／置換] ダイアログを開く
[Ctrl + G]	[SQL 文] ウィンドウ枠内の指定の行に移動する
[Ctrl + H]	実行した SQL の履歴を表示する
[Ctrl + Home]	現在のウィンドウ枠の一番上に移動する
[Ctrl + N]	Interactive SQL ウィンドウの内容をクリアし、現在のファイルを閉じる (存在する場合)。
[Ctrl + O]	ファイルを開く
[Ctrl + P]	[SQL 文] ウィンドウ枠の内容を印刷する。印刷テキストの外観は、Interactive SQL の [オプション] ダイアログで設定できます。「[印刷] タブ」『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。
[Ctrl + Q]	クエリ・エディタを表示する。 クエリ・エディタを使用すると、SQL クエリを作成できます。クエリを構築し終わったら、[OK] をクリックして、[SQL 文] ウィンドウ枠にエクスポートします。
[Ctrl + S]	[SQL 文] ウィンドウ枠の内容を保存する
[Ctrl + V]	選択したテキストを貼り付ける
[Ctrl + X]	選択したテキストを切り取る
[Ctrl + Y]	最後の操作を繰り返す
[Ctrl + Z]	最後の操作を取り消す
[Esc]	[SQL 文] ウィンドウ枠をクリアする
[F1]	ヘルプを表示する
[F2]	結果セット内の選択した値を編集する。ローのカラム間でタブ移動できます。
[F3]	指定したテキストの次の出現箇所を検索する
[F5]	[SQL 文] ウィンドウ枠にあるすべてのテキストを実行する。 この操作は、ツールバーで [SQL 文の実行] をクリックして実行することもできます。

キー	説明
[F7]	[テーブル名のルックアップ] ダイアログを表示する。 このダイアログでテーブルを検索して選択し、[Enter] を押せば、そのテーブル名が [SQL 文] ウィンドウ枠のカーソル位置に入力されます。また、リストでテーブルを選択し、もう一度 [F7] を押すとそのテーブルのカラムが表示されます。この後カラムを選択して [Enter] キーを押せば、そのカラム名が [SQL 文] ウィンドウ枠のカーソル位置に入力されます。
[F8]	[プロシージャ名のルックアップ] ダイアログを表示する。 このダイアログでプロシージャを検索して選択し、[Enter] キーを押せば、そのプロシージャ名が [SQL 文] ウィンドウ枠のカーソル位置に入力されます。
[F9]	[SQL 文] ウィンドウ枠で選択されているテキストを実行する。 テキストが選択されていない場合は、文全体が実行されます。
[F11]	Interactive SQL がデータベースに接続されていない場合に [接続] ダイアログを開く
[F12]	データベースを切断する
[Page Down]	現在のウィンドウ枠を 1 ページ下にスクロールする
[Page Up]	現在のウィンドウ枠を 1 ページ上にスクロールする
[Shift + F5]	[SQL 文] ウィンドウ枠内の文を実行せずに、その文のプランを表示する
[Shift + F10]	フォーカスのある領域のコンテキスト・メニューを表示する。 このキーボード・ショートカットは、領域を右クリックする代わりに使用します。

[SQL 文] ウィンドウ枠にフォーカスがあるときは、以下のキーボード・ショートカットを使用できます。

キー	説明
[Ctrl +]]	カーソルを閉じカッコまで移動する。カッコ、大カッコ、角カッコ、山カッコに一致するときに、このショートカットを使用します。
[Ctrl + Backspace]	カーソルの左の 1 ワードを削除する
[Ctrl + Del]	カーソルの右の 1 ワードを削除する
[Ctrl + G]	[ジャンプ] ダイアログが開き、ジャンプ先の行を指定できる
[Ctrl + L]	現在の行を [SQL 文] ウィンドウ枠から削除してクリップボードに入れる

キー	説明
[Ctrl + Shift +]]	選択範囲を閉じカッコまで拡張する。カッコ、大カッコ、角カッコ、山カッコに一致するときに、このショートカットを使用します。
[Ctrl + Shift + L]	現在の行を削除する
[Ctrl + Shift + U]	選択している内容を大文字に変換する
[Ctrl + U]	選択している内容を小文字に変換する
[F3]	選択したテキストの次の出現箇所を検索する
[Home]	カーソルを現在の行の行頭、または現在の行の最初の単語に移動する
[Shift + F3]	選択したテキストの 1 つ前の出現箇所を検索する
[Shift + Home]	選択範囲を現在の行のテキストの先頭まで拡張する

Interactive SQL オプション

構文 1

SET [**TEMPORARY**] **OPTION** *option-name* = [*option-value*]

構文 2

SET PERMANENT

構文 3

SET

パラメータ

userid: *identifier* または *string*

option-name: *identifier* または *string*

option-value: *string*, *identifier* または *number*

説明

構文 1 は、指定された Interactive SQL オプションを格納します。

構文 2 は、現在のすべての Interactive SQL オプションを格納します。

構文 3 は、現在のオプション設定すべてを表示します。

Interactive SQL のオプション設定はクライアント・コンピュータに保存されます。これらのオプション設定はデータベースには保存されません。

注意
構文 SET [TEMPORARY] OPTION [*userid* | PUBLIC.]*option-name* は、廃止されました。この構文を指定すると、*userid* または PUBLIC キーワードは無視されます。

オプション	値	デフォルト
「auto_commit オプション [Interactive SQL]」 609 ページ	On、Off	Off
「auto_refetch オプション [Interactive SQL]」 610 ページ	On、Off	On
「bell オプション [Interactive SQL]」 610 ページ	On、Off	On
「command_delimiter オプション [Interactive SQL]」 611 ページ	文字列	';'
「commit_on_exit オプション [Interactive SQL]」 612 ページ	On、Off	On
「default_isql_encoding オプション [Interactive SQL]」 612 ページ	文字列	空の文字列
「echo オプション [Interactive SQL]」 613 ページ	On、Off	On
「input_format オプション [Interactive SQL]」 614 ページ	ASCII、DBASE、DBASEII、DBASEIII、EXCEL、FIXED、FOXPRO、LOTUS	ASCII
「isql_command_timing オプション [Interactive SQL]」 614 ページ	On、Off	On
「isql_escape_character オプション [Interactive SQL]」 615 ページ	文字	'¥'
「isql_field_separator オプション [Interactive SQL]」 616 ページ	文字列	','
「isql_maximum_displayed_rows オプション [Interactive SQL]」 617 ページ	All または負でない整数	500
「isql_plan オプション [Interactive SQL]」 617 ページ	Short、Long、GraphicalLowDetail、GraphicalHighDetail	Graphical
「isql_print_result_set オプション [Interactive SQL]」 618 ページ	Last、All、None	Last

オプション	値	デフォルト
「isql_quote オプション [Interactive SQL]」 619 ページ	文字列	'
「isql_show_multiple_result_sets [Interactive SQL]」 619 ページ	On、Off	Off
「nulls オプション [Interactive SQL]」 620 ページ	文字列	'(NULL)'
「on_error オプション [Interactive SQL]」 620 ページ	Stop、Continue、Prompt、Exit、Notify_Continue、Notify_Stop、Notify_Exit	Prompt
「output_format オプション [Interactive SQL]」 621 ページ	ASCII、DBASEII、DBASEIII、EXCEL、FIXED、FOXPRO、HTML、LOTUS、SQL、XML	ASCII
「output_length オプション [Interactive SQL]」 622 ページ	整数	0
「output_nulls オプション [Interactive SQL]」 623 ページ	文字列	空の文字列
「truncation_length オプション [Interactive SQL]」 623 ページ	整数	256

参照

- 「SET OPTION 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』

auto_commit オプション [Interactive SQL]

機能

各文の後で COMMIT が実行されるかどうかを制御します。

指定可能な値

On、Off

デフォルト

Off

説明

auto_commit が On の場合、各文の処理が成功した後でデータベースの COMMIT が実行されます。

デフォルトでは、COMMIT または ROLLBACK が実行されるのは、ユーザが COMMIT 文または ROLLBACK 文を発行するか、自動コミットを行う SQL 文 (CREATE TABLE 文など) を発行したときだけです。

参照

- 「SET OPTION 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

auto_refetch オプション [Interactive SQL]

機能

削除、更新、挿入後にクエリ結果が再度フェッチされるかどうかを制御します。

指定可能な値

On、Off

デフォルト

On

説明

auto_refetch が On の場合、Interactive SQL の [結果] ウィンドウ枠の [結果] タブに表示される現在のクエリ結果は、INSERT 文、UPDATE 文、または DELETE 文の後でデータベースから再フェッチされます。クエリの複雑さによって、時間のかかるものもあります。このため、これをオフにすることもできます。

参照

- 「SET OPTION 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

bell オプション [Interactive SQL]

機能

エラーが起こったときにベルを鳴らすかどうかを制御します。

指定可能な値

On、Off

デフォルト

On

説明

このオプションは好みに応じて設定します。

参照

- 「SET OPTION 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

command_delimiter オプション [Interactive SQL]

機能

Interactive SQL で文の終わりを示す文字列を設定します。

指定可能な値

文字列

デフォルト

セミコロン (;)

説明

通常は、コマンド・デリミタを変更する必要はありません。セミコロンのままにしてください。

文のデリミタとしてセミコロンなどの文字列を使用する代わりに、セパレータ go を独立した行の先頭に配置することもできます。「バッチの概要」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

独立した行に指定された go は、command_delimiter オプションの値にかかわらず常にコマンド・デリミタとして認識されます。

command_delimiter オプションには任意の文字列を指定できます。ただし、次のような制限があります。

- & (アンパサンド)、* (アスタリスク)、@ (アットマーク)、: (コロン), . (ピリオド)、= (等号)、((左カッコ)、) (右カッコ)、または | (縦線) が含まれているデリミタには、それ以外の文字を追加することはできません。たとえば、* は有効なデリミタですが、** はデリミタとして無効です。
- 既存のキーワードをコマンド・セパレータとして使用しないでください。
- コマンド・デリミタには、数字、文字、句読点などの任意の文字列を使用できますが、埋め込みブランクを含めることはできません。また、セミコロンは、先頭文字としてだけ含めることができます。

  コマンド・デリミタとして設定されている文字列が、識別子として有効な文字で始まる場合は、前にスペースを付けてください。コマンド・デリミタでは、大文字と小文字が区別されます。新しいコマンド・デリミタは一重引用符で囲んでください。コマンド・デリミタがセミコロン (デフォルト) の場合、セミコロンの前にスペースを入れる必要はありません。

参照

- 「Interactive SQL ユーティリティ (dbisql)」 672 ページ
- 「SET OPTION 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

例

次の例は、コマンド・デリミタをチルダに設定します。

```
SET OPTION command_delimiter='~';
MESSAGE 'hello'~
```


Interactive SQL の -d オプションを使用してコマンド・デリミタを設定することもできます。この場合、*.sql* ファイルに SET OPTION command_delimiter 文を記述する必要はありません。たとえば、スクリプト・ファイル *test.sql* で、チルダ (~) をコマンド・デリミタとして使用する場合は、次のようになります。

```
dbisql -d "~" test.sql
```

commit_on_exit オプション [Interactive SQL]

機能

Interactive SQL が切断、終了されるときの動作を制御します。

指定可能な値

On、Off

デフォルト

On

説明

Interactive SQL を終了するときに COMMIT か ROLLBACK を行うかを制御します。commit_on_exit を On に設定すると、COMMIT が行われます。

参照

- 「SET OPTION 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』

default_isql_encoding オプション [Interactive SQL]

機能

READ、INPUT、OUTPUT 文で使用されるコード・ページを指定します。

指定可能な値

識別子または文字列

デフォルト

システム・コード・ページ (空の文字列) を使用します。

スコープ

現在の接続の間、テンポラリ・オプションとしてのみ設定できます。

説明

このオプションは、ファイルを読み込みまたは書き込みするときに使用するコード・ページを指定するのに使用されます。これを永続的に設定することはできません。デフォルトのコード・ページは、実行しているプラットフォームのデフォルト・コード・ページです。英語版 Windows コンピュータでは、デフォルトのコード・ページは 1252 です。

Interactive SQL は、次のような特定の INPUT、OUTPUT、または READ 文に使用されるコード・ページを判断します。ここで、リストの上位で生成されたコード・ページ値は下位で生成されたものより優先されます。

- INPUT 文、OUTPUT 文、または READ 文の ENCODING 句に指定されたコード・ページ
- DEFAULT_SQL_ENCODING オプションで指定されたコード・ページ (このオプションが設定されている場合)
- Interactive SQL の開始時に -codepage で指定されたコード・ページ
- Interactive SQL が実行されているコンピュータのデフォルトのコード・ページ

コード・ページと文字セットの詳細については、「国際言語と文字セットのタスク」 354 ページを参照してください。

参照

- 「READ 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「INPUT 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「OUTPUT 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「文字セット、エンコード、照合の概要」 341 ページ
- 「SET OPTION 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

例

コード化を UTF-16 (ユニコード・ファイルの読み込み用) に設定します。

```
SET OPTION default_isql_encoding = 'UTF-16';
```

echo オプション [Interactive SQL]

機能

文を実行する前にそれをログ・ファイルへエコーするかどうかを制御します。

指定可能な値

On、Off

デフォルト

On

説明

このオプションは、READ 文を使用して Interactive SQL コマンド・ファイルを実行するのに最も便利です。このオプションを有効にするには、ロギングをオンにする必要があります。「START LOGGING 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

参照

- 「SET OPTION 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

input_format オプション [Interactive SQL]

機能

INPUT 文が予期するデフォルト・データ・フォーマットを設定します。

指定可能な値

文字列 (指定可能な値は以下を参照)

デフォルト

ASCII

説明

使用できる入力フォーマットは、次のとおりです。

- **ASCII** 入力行は ASCII 文字であり、1 行あたり 1 つのローで構成され、値はカンマで区切られているものとみなされます。アルファベットの文字列をアポストロフィ (一重引用符) または二重引用符で囲むことができます。カンマを含む文字列は、一重引用符または二重引用符のどちらかで囲む必要があります。一重か二重引用符を使用している場合、文字列内で使用するには引用符を 2 つ重ねてください。オプションで DELIMITED BY 句を指定すると、デフォルトのカンマ (,) 以外のデリミタを使用できます。

  他の 3 つの特別なシーケンスも認識できます。2 つの文字 ¥n は改行文字を表し、¥¥ は 1 つの円記号 (¥) を表し、シーケンス ¥xDD (DD は文字の 16 進表現) は 16 進コード DD の文字を表します。
- **DBASE** このファイルは、dBASE II または dBASE III フォーマットです。Interactive SQL はどちらのフォーマットであるかを、ファイル内の情報に基づいて決定しようとします。
- **DBASEII** このファイルは dBASE II フォーマットです。
- **DBASEIII** このファイルは dBASE III フォーマットです。
- **EXCEL** 入力ファイルは Microsoft Excel 2.1 のフォーマットです。
- **FIXED** 入力行は固定フォーマットです。
- **FOXPRO** このファイルは FoxPro フォーマットです。
- **LOTUS** このファイルは Lotus WKS フォーマットのワークシートです。INPUT は Lotus WKS フォーマット・ワークシートの最初のローがカラム名から構成されると見なします。

参照

- 「INPUT 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「SET OPTION 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

isql_command_timing オプション [Interactive SQL]

機能

SQL 文の実行時間を記録するかどうかを制御します。

指定可能な値

On、Off

デフォルト

On

説明

このブール・オプションでは、SQL 文の実行時間を記録するかどうかを制御します。このオプションを On に設定すると、SQL 文を実行した後に、実行時間が [メッセージ] タブに表示されます。Off に設定すると、実行時間は表示されません。

このオプションは、[オプション] ダイアログの [メッセージ] タブで設定することもできます。

参照

- 「SET OPTION 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

isql_escape_character オプション [Interactive SQL]

機能

ASCII ファイルにエクスポートされたデータに印刷不能な文字が含まれていた場合に、その代わりに使用するエスケープ文字を制御します。

指定可能な値

任意の 1 文字

デフォルト

円記号 (¥)

説明

Interactive SQL が改行など印刷不能な文字を含む文字列をエクスポートする場合、印刷不能な文字は 16 進数フォーマットに変換され、その文字の前にはエスケープ文字が付加されます。OUTPUT 文に ESCAPE CHARACTER 句がない場合に、この設定で指定した文字が出力で使用されます。この設定は、ASCII ファイルにエクスポートする場合にだけ使用されます。

参照

- 「isql_quote オプション [Interactive SQL]」 619 ページ
- 「SET OPTION 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

例

改行が埋め込まれた 1 つの文字列値を含む表を作成します (INSERT 文に "¥n" で示されます)。次に、# 記号をエスケープ文字として使用して、データを c:¥escape.txt へエクスポートします。


```
CREATE TABLE escape_test( text varchar(10 ) );
INSERT INTO escape_test VALUES( 'one¥ntwo' );
SET OPTION isql_escape_character='#';
SELECT * FROM escape_test;
OUTPUT TO c:¥escape.txt FORMAT ASCII;
```

このコードを実行すると、次のデータが *escape.txt* に書き込まれます。

```
'one#x0Atwo'
```

ポンド記号 (#) はエスケープ文字であり、**x0A** は **¥n** の 16 進表現です。

先頭と末尾の文字 (この場合は一重引用符) は、isql_quote の設定によって異なります。

isql_field_separator オプション [Interactive SQL]

機能

ASCII ファイルにエクスポートするデータの値を区切るのに使用される、デフォルトの文字列を制御します。

指定可能な値

文字列

デフォルト

カンマ (,)

説明

ASCII ファイルにエクスポートするデータの値を区切るのに使用される、デフォルトの文字列を制御します。OUTPUT 文に DELIMITED BY 句が含まれていない場合は、この設定値が使用されます。

参照

- 「isql_quote オプション [Interactive SQL]」 619 ページ
- 「SET OPTION 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

例

c:¥Employees.txt にエクスポートするデータのフィールド・セパレータとしてコロンを指定します。

```
SET OPTION isql_field_separator=':';
SELECT Surname, GivenName FROM Employees WHERE EmployeeID < 150;
OUTPUT TO c:¥Employees.txt FORMAT ASCII;
```

このコードを実行すると、次のデータが *Employees.txt* に書き込まれます。

```
'Whitney':'Fran'

'Cobb':'Matthew'

'Chin':'Philip'

'Jordan':'Julie'
```


先頭と末尾の文字 (この場合は一重引用符) は、isql_quote の設定によって異なります。

isql_maximum_displayed_rows オプション [Interactive SQL]

機能

Interactive SQL で [結果] ウィンドウ枠に表示されるローの最大数を指定します。

指定可能な値

ALL または負でない整数

デフォルト

500

説明

このオプションでは、[結果] ウィンドウ枠に表示されるローの最大数を指定できます。このオプションの値は、Interactive SQL の [オプション] ダイアログでも設定できます。

参照

- 「SET OPTION 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

isql_plan オプション [Interactive SQL]

機能

文の実行後に Interactive SQL の [結果] ウィンドウ枠の [プラン] タブに表示されるアクセス・プランのタイプを制御します。

指定可能な値

SHORT、LONG、GRAPHICAL、GRAPHICALLOWDETAIL、GRAPHICALHIGHDETAIL

デフォルト

GRAPHICAL

説明

このオプションを使用して、SELECT、INSERT、UPDATE、DELETE 文用に表示されるアクセス・プランのタイプを指定できます。オプティマイザが提供するプランの詳細レベルを設定するために、次の値のいずれかを選択できます。

- **SHORT** アクセス・プランに関して、基本的な情報が表示されます。
- **LONG** アクセス・プランに関して、詳細な情報が表示されます。
- **GRAPHICAL** 統計情報なしのグラフィカルなプランが表示されます。
- **GRAPHICALLOWDETAIL** クエリのルート・ノードに関する、クエリのツリー状の図が統計情報付きで表示されます。

- **GRAPHICALHIGHDETAIL**　クエリのツリー状の図と、選択されたクエリの部分で使用されているリソースを示す統計が表示されます。

プランは、[プラン] タブをクリックしたときだけ計算されます。

Interactive SQL の [オプション] ダイアログの [プラン] タブに、プラン・タイプを指定することもできます。

表示されるアクセス・プランは、文が実行されたときに使用されたプランと同じではない場合があることに注意してください。プランは、SQL 文が実行された後、オプティマイザが新しい情報を使用できる時点で取り出されます。つまり、オプティマイザが文を実行するのに使用したものとは異なるプランを戻す場合があります。

参照

- 「SET OPTION 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』

isql_print_result_set オプション [Interactive SQL]

機能

.sql ファイルを実行したときに出力される結果セットを指定します。

isql_print_result_set オプションは、コマンド・ウィンドウで Interactive SQL (dbisql) ユーティリティを実行する場合 (たとえば、*.sql* ファイルの実行時) のみ有効です。

指定可能な値

LAST、ALL、NONE

デフォルト

LAST

説明

このオプションでは、*.sql* ファイルを実行したときに出力する結果セットを指定できます。

次のいずれかの出力オプションを選択できます。

- **LAST**　ファイル内の最後の文の結果セットを出力します。
- **ALL**　結果セットを返すファイル内のすべての文の結果セットを出力します。
- **NONE**　結果セットを出力しません。

このオプションは、Interactive SQL がウィンドウ・モードで実行しているときは機能しませんが、ウィンドウ・モードでもオプションの表示と設定は行えます。[ツール] – [オプション] を選択します。[結果] タブの [コンソール・モード] 領域で適切なアクションを選択します。

参照

- 「SET OPTION 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』

isql_quote オプション [Interactive SQL]

機能

ASCII ファイルにエクスポートするデータの文字列すべての先頭と最後に付く、デフォルトの文字列を制御します。

指定可能な値

文字列

デフォルト

一重引用符 (')

説明

ASCII ファイルにエクスポートするデータの文字列すべての先頭と最後に付く、デフォルトの文字列を制御します。OUTPUT 文に QUOTE 句が含まれていない場合は、この設定値がデフォルトで使用されます。

参照

- 「isql_field_separator オプション [Interactive SQL]」 616 ページ
- 「SET OPTION 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

例

すべての文字列の先頭と最後に付くデフォルト文字列を二重引用符に変更するには、次のように指定します。

```
SET OPTION isql_quote='"';
SELECT Surname, GivenName FROM Employees WHERE EmployeeID < 150;
OUTPUT TO c:¥Employees.txt FORMAT ASCII;
```

このコードを実行すると、次のデータが *Employees.txt* に書き込まれます。

"Whitney", "Fran"

"Cobb","Matthew"

"Chin","Philip"

"Jordan","Julie"

区切り文字 (この場合はカンマ) は、isql_field_separator の設定によって異なります。

isql_show_multiple_result_sets [Interactive SQL]

機能

Interactive SQL で [結果] ウィンドウ枠に複数の結果セットを表示できるようにするかどうかを指定します。

指定可能な値

ON、OFF

デフォルト

OFF

説明

複数の SELECT 文を返すプロシージャを実行したときに複数の結果セットを [結果] ウィンドウ枠に表示するには、このオプションを ON に設定します。

各結果セットは [結果] ウィンドウ枠の個別のタブに表示されます。デフォルトでは、Interactive SQL では複数の結果セットは表示されません。このオプションの設定は、Interactive SQL をコンソール (コマンドプロンプト) モードで実行する場合にも適用されます。

参照

- 「SET OPTION 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

nulls オプション [Interactive SQL]

機能

Interactive SQL で結果を表示するときにデータベース内の NULL 値をどのように表示するかを指定します。

指定可能な値

文字列

デフォルト

(NULL)

説明

このオプションは好みに応じて設定します。結果セットをファイルに保存する場合は、このオプションの値は適用されません。このオプションと同じ設定をファイルに保存する結果セットに適用するには、output_nulls オプションを使用します。

参照

- 「SET OPTION 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「output_nulls オプション [Interactive SQL]」 623 ページ

on_error オプション [Interactive SQL]

機能

Interactive SQL の文を実行中にエラーが起こった場合の動作を制御します。

指定可能な値

文字列 (指定可能な値は以下を参照)

デフォルト

Prompt

説明

文の実行中にエラーが発生した場合の動作を次のように制御します。

- **Stop** Interactive SQL が文の実行を停止します。
- **Prompt** Interactive SQL は、ユーザに継続したいかどうかを確認するプロンプトを表示します。
- **Continue** エラーは無視され、Interactive SQL は文の実行を継続します。
- **Exit** Interactive SQL は終了します。
- **Notify_Continue** エラーがレポートされますが、ユーザは継続させるために [Enter] キーを押すか、[OK] をクリックするよう要求されます。
- **Notify_Stop** エラーがレポートされますが、ユーザは実行を停止するために [Enter] キーを押すか、[OK] をクリックするよう要求されます。
- **Notify_Exit** エラーがレポートされますが、ユーザは Interactive SQL を終了するために [Enter] を押すか、[OK] をクリックするよう要求されます。

.sql ファイルを実行する場合は、Stop と Exit のどちらに設定しても同じ結果となります。これらの値のいずれかを指定すると、Interactive SQL が終了します。

参照

- 「SET OPTION 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』

output_format オプション [Interactive SQL]

機能

ファイルにリダイレクトされる SELECT 文で検索したデータや、OUTPUT 文を使用した出力のデフォルトの出力フォーマットを設定します。

指定可能な値

文字列 (指定可能な値は以下を参照)

デフォルト

ASCII

説明

有効な出力フォーマットは次のとおりです。

- **ASCII**　出力は ASCII フォーマット・ファイルであり、1 行あたり 1 つのローで構成されます。すべての値をカンマで区切り、文字列をアポストロフィ (一重引用符) で囲みます。デリミタと引用符文字列は、DELIMITED BY と QUOTE 句を使って変更できます。QUOTE 句に All が指定されている場合は、文字列だけでなく、すべての値が引用符で囲まれます。

  他の 3 つの特別なシーケンスも使用できます。2 つの文字 ¥n は改行文字を表し、¥¥ は 1 つの円記号 (¥) を表し、シーケンス ¥xDD は 16 進コード DD の文字を表します。

- **DBASEII**　出力は、ファイルの最上部にカラム定義がある dBASE II フォーマット・ファイルです。最大 32 カラムを出力できることに注意してください。カラム名は 11 文字にトランケートされ、各カラムのデータの各ローは 255 文字にトランケートされます。

- **DBASEIII**　出力は、ファイルの最上部にカラム定義がある dBASE III フォーマット・ファイルです。最大 128 カラムを出力できることに注意してください。カラム名は 11 文字にトランケートされ、各カラムのデータの各ローは 255 文字にトランケートされます。

- **EXCEL**　この出力は Excel 2.1 のワークシートです。ワークシートの最初のローには、カラム・ラベル (または、ラベルが定義されていない場合はカラム名) があります。2 つ目以降のワークシート・ローには、実際のテーブル・データがあります。

- **FIXED**　出力は、それぞれのカラムが固定幅を持つ固定フォーマットです。それぞれのカラムの幅は COLUMN WIDTH 句を使って指定できます。この句を省略した場合、各カラムの幅はカラムのデータ型から計算され、そのデータ型の値を保持するのに十分な大きさになります。カラムの見出しはこのフォーマット中では出力されません。

- **FOXPRO**　出力は、ファイルの最上部にカラム定義がある FoxPro フォーマット・ファイルです。最大 128 カラムを出力できることに注意してください。カラム名は 11 文字にトランケートされ、各カラムのデータの各ローは 255 文字にトランケートされます。

- **HTML**　出力は HTML フォーマットです。

- **LOTUS**　出力は、Lotus WKS フォーマットのワークシートです。カラム名をワークシートの最初のローとして入れます。(Lotus 1-2-3 のような) 他のソフトウェアがロードできる Lotus WKS フォーマット・ワークシートの最大サイズに、一定の制限があることに注意してください。Interactive SQL のファイル・サイズには制限はありません。

- **SQL**　出力は、テーブル内の情報を再作成するのに必要な Interactive SQL の INPUT 文です。

- **XML**　この出力は、UTF-8 でエンコードされ、DTD が埋め込まれた XML ファイルです。バイナリ値は、2 桁の 16 進数文字列として表されるバイナリ・データとして CDATA ブロック内にエンコードされます。

参照

- 「SET OPTION 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

output_length オプション [Interactive SQL]

機能

Interactive SQL が、外部ファイルに情報をエクスポートするときに使用するカラム値の長さを制御します。

指定可能な値

負でない整数

デフォルト

0 (トランケーションなし)

説明

このオプションは、Interactive SQL が外部ファイルにデータをエクスポートするときに使用するカラム値の最大長を制御します (OUTPUT 文で出力リダイレクションを使用)。このオプションは、ASCII、HTML、SQL の出力フォーマットだけに影響します。

参照

- 「SET OPTION 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』

output_nulls オプション [Interactive SQL]

機能

NULL 値をどのようにエクスポートするかを制御します。

指定可能な値

文字列

デフォルト

空の文字列

説明

このオプションは、NULL 値をどのように OUTPUT 文で記述するかを制御します。結果セットで NULL 値が見つかった場合、NULL 値の代わりにこのオプションで設定された文字列が返されます。このオプションは、ASCII、HTML、FIXED、EXECL、SQL の出力フォーマットだけに影響します。

参照

- 「SET OPTION 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』

truncation_length オプション [Interactive SQL]

機能

表示内容を画面内に収めるために、幅の広いカラムのトランケーションを制御します。

指定可能な値

整数

デフォルト

256

説明

truncation_length オプションは、表示されるカラムの長さを制限します。単位は文字です。値 0 はカラムがトランケートされないことを意味します。トランケーションのデフォルト値は 256 です。

参照

- 「SET OPTION 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』

テキスト補完の使用

Interactive SQL と Sybase Central は、オブジェクト名を入力できるテキスト補完オプションを提供します。テキスト補完オプションを設定して、テーブル、ビュー、カラム、ストアド・プロシージャ、システム関数の名前を入力できます。

SELECT、INSERT、UPDATE、DELETE、DESCRIBE 文では、入力しているところに関連すると考えられる候補のリストが提示されます。たとえば、次の SQL 文について考えてみます。

```
SELECT EmployeeID FROM Employees as e WHERE e.EmployeeID>=20;
```

SELECT を入力した後にテキスト補完ウィンドウを開くと、Employees テーブルのカラム名、ストアド・プロシージャ、SQL 関数を含むリストが表示されます。

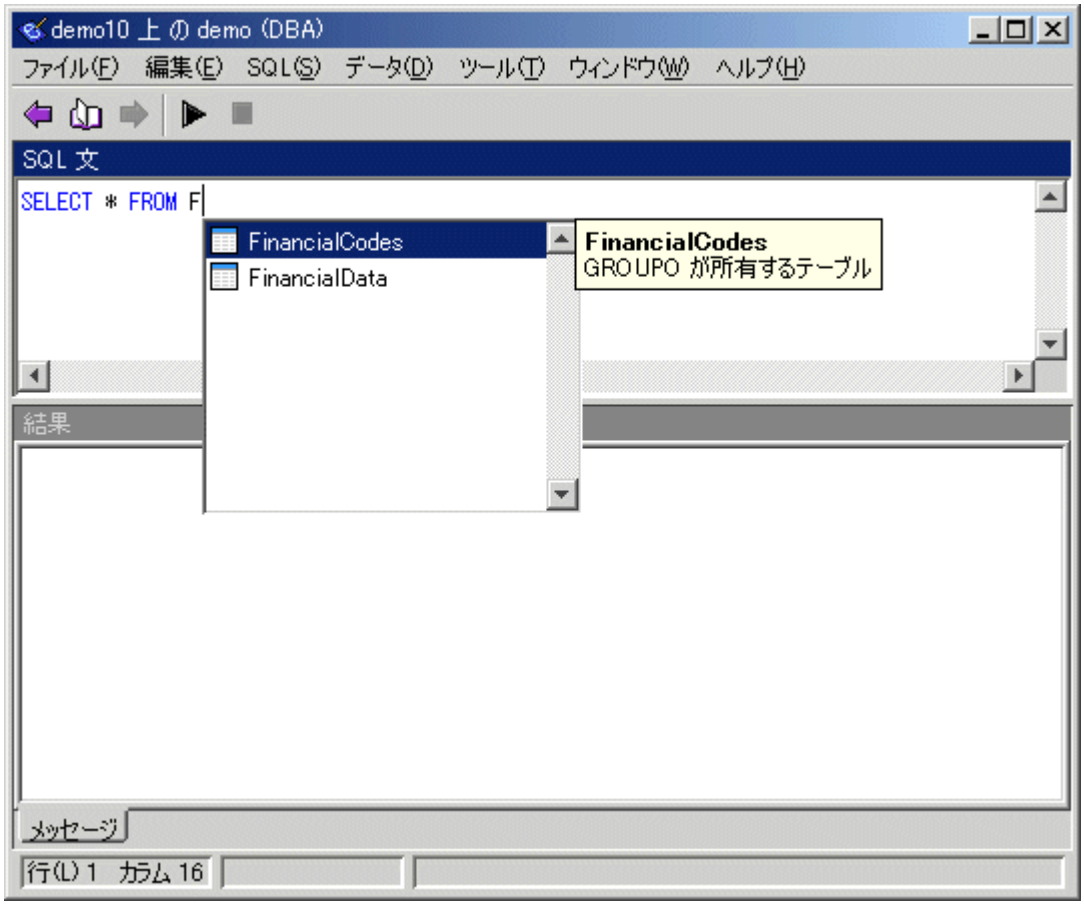
FROM を入力した後にテキスト補完ウィンドウを開くと、テーブルとストアド・プロシージャのみを含むリストが表示されます。

WHERE の 2 番目の e を入力した後にテキスト補完ウィンドウを開くと、エイリアスが e のテーブルにあるカラムのみを含むリストが表示されます。

♦ **テキスト補完を使用するには、次の手順に従います。**

1. Interactive SQL の [SQL 文] ウィンドウ枠で、データベース・オブジェクトの最初の文字を入力します。
2. [Ctrl] キーとスペース・キー、または [Ctrl + Shift] キーとスペース・キーを押します。

   ウィンドウが開き、今までに入力した文字で始まるデータベース・オブジェクトの名前がリストされます。次の例では、文字 F で始まるすべてのデータベース・オブジェクトが表示されます。



使用したいオブジェクト名が見つからない場合は、[Tab] キーを押して、データベース・オブジェクトの完全なリストを表示します (完全なリストは、ユーザが設定したフィルタリング オプションに基づいて表示されます。デフォルトでは、すべてのデータベース・オブジェクトが表示されます)。

3. リストからオブジェクト名を選択し、[Enter] キーを押します。

   [SQL 文] ウィンドウ枠にオブジェクト名が表示されます。

テキスト補完の設定は、Interactive SQL の [オプション] ダイアログから変更できます。Sybase Central のテキスト・エディタ・ウィンドウを使用することも可能です。

テキスト補完キーボード・ショートカット

テキスト補完リストが開いているときに、以下のキーボード・ショートカットを使用できます。

キー	説明
[Ctrl + C]	テキスト補完リストにカラムのみを表示する
[Ctrl + F]	テキスト補完リストに SQL 関数のみを表示する
[Ctrl + P]	テキスト補完リストにストアド・プロシージャと関数のみを表示する
[Ctrl + S]	リストの内容を変更してシステム・オブジェクトを表示する、または非表示にする
[Ctrl + Shift] キーとスペース・キー	テキスト補完ウィンドウを開く。[Ctrl] キーとスペース・キーを使用して、テキスト補完ウィンドウを開くこともできます。
[Ctrl + T]	テキスト補完リストにテーブルのみを表示する
[Ctrl + V]	テキスト補完リストにビューのみを表示する
[Esc]	テキストを追加しないでテキスト補完ウィンドウを閉じる
[Tab]	すべてのデータベース・オブジェクト名のリストと、今までに入力した文字に一致する名前を持つデータベース・オブジェクト名のリストを切り替える
*	テーブルの場合：カラムのカンマ区切りのリスト (データ型を含む) を挿入する ストアド・プロシージャの場合：プロシージャ名を挿入し、続けてパラメータ名とデータ型のカンマ区切りのリストを挿入する
+	テーブルの場合：カラムのカンマ区切りのリストを挿入する ストアド・プロシージャの場合：プロシージャ名を挿入し、続けてパラメータ名のカンマ区切りのリストを挿入する
"	[引用符付きの識別子] オプションの設定に関係なく、引用符で囲んで名前の入力を完了する。「quoted_identifier オプション [互換性]」 487 ページを参照してください。

高速ランチャの使用

高速ランチャは、Sybase Central および Interactive SQL の起動時間を短縮するように設計されています。高速ランチャを有効にすると、ユーザのログイン時に高速ランチャ・プロセス (Sybase Central の *scjview.exe* と Interactive SQL の *dbisqlg.exe*) が起動します。高速ランチャは Windows のみで使用可能です。

Sybase Central または Interactive SQL で高速ランチャを無効にしても、高速ランチャ・プロセスはログアウトするかコンピュータを再起動するまで終了しません。

高速ランチャの設定

高速ランチャは、ユーザのコンピュータ上の TCP/IP ポートを使用します。別のプログラムがこのポートを使用している場合は、高速ランチャによって使用されるポート番号を変更できます。

休止タイマに指定した時間に高速ランチャが使用されない場合、高速ランチャは停止します。これにより、他のアプリケーション用のメモリが解放されます。デフォルトでは、休止タイマは停止しないようになっています。

♦ Interactive SQL の高速ランチャを設定するには、次の手順に従います。

1. Interactive SQL を開きます。
2. [ツール] – [オプション] を選択します。

   [オプション] ダイアログが表示されます。
3. [オプション] ダイアログの [一般] タブで、[設定] をクリックします。

   [Interactive SQL 高速ランチャの設定] ダイアログが表示されます。
4. 必要に応じて、高速ランチャのポートと休止タイマを設定します。
5. [OK] をクリックします。
6. [OK] をクリックします。

♦ Sybase Central の高速ランチャを設定するには、次の手順に従います。

1. Sybase Central を開きます。
2. [ツール] – [オプション] を選択します。

   [オプション] ダイアログが表示されます。
3. [オプション] ダイアログの [一般] タブで、[設定] をクリックします。

   [Sybase Central 高速ランチャの設定] ダイアログが表示されます。
4. 必要に応じて、高速ランチャのポートと休止タイマを設定します。
5. [OK] をクリックします。
6. [OK] をクリックします。

SQL Anywhere コンソール・ユーティリティ

SQL Anywhere コンソール・ユーティリティは、データベース・サーバ接続の管理機能とモニタリング機能を提供します。

SQL Anywhere ユーティリティは、Windows CE、AIX、HP-UX、HP-UX Itanium、Linux Itanium を除き、サポートされるすべてのプラットフォームで使用可能です。これらのプラットフォームでは、接続、データベース、サーバのプロパティを使用して情報を取得したり、SQL Anywhere コンソールをサポートするオペレーティング・システム (Windows、Mac OS X、Linux など) を実行するコンピュータからサーバをモニタしたりすることができます。

DBA 権限を持たないユーザが SQL Anywhere Console ユーティリティに接続すると、DBA 権限を必要とするすべての機能は無効になります。

SQL Anywhere コンソール・ユーティリティを実行したり使用したりする方法の詳細については、「SQL Anywhere コンソール・ユーティリティ (dbconsole)」 718 ページを参照してください。

SQL Anywhere コンソール・ユーティリティの起動

♦ SQL Anywhere コンソール・ユーティリティを起動するには、次の手順に従います (コマンド・プロンプトの場合)。

- コマンド・プロンプトから dbconsole コマンドを実行します。

  データベースへの接続パラメータを指定する -c オプションを省略するなど、接続パラメータの指定が不十分であると、[接続] ダイアログが表示されます。そのダイアログに、データベースに対する接続情報を入力できます。

  サポートされるオプションの詳細については、「SQL Anywhere コンソール・ユーティリティ (dbconsole)」 718 ページを参照してください。

  次のコマンドは、Interactive SQL を起動し、サンプル・データベースに接続します。

  ```
  dbconsole -c "UID=DBA;PWD=sql;DSN=SQL Anywhere 10 Demo"
  ```

Linux のデスクトップ・アイコンをサポートするバージョンの Linux を使用している場合で、SQL Anywhere 10 をインストールするときにこれらのアイコンをインストールするように選択したときは、次の手順を使用できます。

♦ SQL Anywhere コンソール・ユーティリティを起動するには、次の手順に従います (Linux デスクトップ・アイコンの場合)。

1. デスクトップで [SQL Anywhere 10] フォルダを開きます。
2. [DBConsole] をダブルクリックします。

   SQL Anywhere コンソール・ユーティリティが開き、[接続] ダイアログが表示されます。
3. [接続] ダイアログで、データベースの接続情報を入力します。

注意
以降の手順は、SQL Anywhere ユーティリティのソースを指定済みであることを前提としています。「UNIX と Mac OS X での環境変数の設定」 306 ページを参照してください。

♦ **SQL Anywhere コンソール・ユーティリティを起動するには、次の手順に従います (UNIX コマンド・ラインの場合)。**

1. ターミナル・セッションで次のコマンドを入力します。

   ```
   dbconsole
   ```

   SQL Anywhere コンソール・ユーティリティが開き、[接続] ダイアログが表示されます。
2. [接続] ダイアログで、データベースの接続情報を入力します。

SQL Anywhere コンソール・ユーティリティのメイン・ウィンドウ

SQL Anywhere コンソール・ユーティリティは、3 つのウィンドウ枠で構成されます。

- **接続**　現在のデータベース接続に関する情報を表示します。
- **プロパティ**　現在実行中のデータベースとデータベース・サーバに関する情報を表示します。
- **メッセージ**　データベース・サーバ・メッセージを表示します。

[オプション] ダイアログを使用して、各ウィンドウ枠に表示される情報を設定できます。

♦ **[接続] ウィンドウ枠の内容をカスタマイズするには、次の手順に従います。**

1. SQL Anywhere コンソール・ユーティリティで、[ファイル] – [オプション] を選択します。
   [オプション] ダイアログが表示されます。
2. 左ウィンドウ枠で [接続ビューワ] をクリックします。
3. [接続ビューワ] タブで、[接続] ウィンドウ枠に表示するプロパティを選択します。

♦ **[プロパティ] ウィンドウ枠の内容をカスタマイズするには、次の手順に従います。**

1. SQL Anywhere コンソール・ユーティリティで、[ファイル] – [オプション] を選択します。
   [オプション] ダイアログが表示されます。
2. 左ウィンドウ枠で [プロパティ・ビューワ] をクリックします。
3. [プロパティ・ビューワ] タブで、[プロパティ] ウィンドウ枠に表示するデータベース・プロパティとサーバ・プロパティを選択します。

♦ **[メッセージ] ウィンドウ枠の内容をカスタマイズするには、次の手順に従います。**

1. SQL Anywhere コンソール・ユーティリティで、[ファイル] – [オプション] を選択します。

[オプション] ダイアログが表示されます。

2. 左ウィンドウ枠で [メッセージ・ビューワ] をクリックします。
3. [メッセージ・ビューワ] タブで、目的のオプションを選択します。

ソフトウェア更新のチェック

EBF などの更新やメンテナンス・リリースが使用可能になったときに通知されるように SQL Anywhere を設定できます。デフォルトでは、SQL Anywhere はソフトウェア更新をチェックしません。

更新の自動チェック

Sybase Central、Interactive SQL、SQL Anywhere コンソール・ユーティリティ (dbconsole) には、SQL Anywhere によるソフトウェア更新チェックの有無とその頻度を制御する、更新チェッカを設定する手段が用意されています。

♦ **更新チェッカを設定するには、次の手順に従います (Sybase Central の場合)。**

1. [ヘルプ] – [SQL Anywhere 10] – [更新チェッカの設定] を選択します。

   [オプション] ダイアログの [更新のチェック] タブが表示されます。
2. 更新チェッカ設定を必要に応じて指定します。

♦ **更新チェッカを設定するには、次の手順に従います (Interactive SQL の場合)。**

1. [ツール] – [オプション] を選択します。

   [オプション] ダイアログが表示されます。
2. [更新のチェック] をクリックします。
3. 更新チェッカ設定を必要に応じて指定します。

♦ **更新チェッカを設定するには、次の手順に従います (SQL Anywhere コンソール・ユーティリティの場合)。**

1. [ファイル] – [オプション] を選択します。

   [オプション] ダイアログが表示されます。
2. [更新のチェック] をクリックします。
3. 更新チェッカ設定を必要に応じて指定します。

更新の手動チェック

SQL Anywhere ソフトウェアの更新は、次のいずれかの手順でいつでもチェックできます。

- **[スタート] メニュー**　[スタート] – [プログラム] – [SQL Anywhere 10] – [更新のチェック] を選択します。
- **Sybase Central**　[ヘルプ] – [SQL Anywhere 10] – [更新チェッカの設定] を選択します。
- **Interactive SQL**　[ヘルプ] – [更新のチェック] を選択します。

- **SQL Anywhere コンソール・ユーティリティ (dbconsole)**　[ヘルプ] – [更新のチェック] を選択します。
- **SQL Anywhere サポート・ユーティリティ (dbsupport)**　次のコマンドを発行します。

```
dbsupport -iu
```

参照

- 「SQL Anywhere のエラー・レポート」 56 ページ
- 「SQL Anywhere サポート・ユーティリティ (dbsupport)」 722 ページ

第 16 章

データベース管理ユーティリティ

目次

管理ユーティリティの概要

SQL Anywhere には、データベース管理タスクを行う一連のユーティリティ・プログラムが付属しています。各ユーティリティには、Sybase Central または Interactive SQL からアクセスするか、コマンド・プロンプトでアクセスできます。

管理ユーティリティでは、一連のレジストリ・エントリまたは *.ini* ファイルを使用します。「レジストリと INI ファイル」 328 ページを参照してください。

データベース・ファイル管理文

一連の SQL 文を使用すると、管理ユーティリティが実行するタスクの一部を実行できます。「SQL 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

参照

- 「Sybase Central」 570 ページ
- 「Interactive SQL」 585 ページ

設定ファイルの使用

SQL Anywhere で提供されている多くのユーティリティでは、設定ファイルにコマンド・ライン・オプションを保存できます。オプションの拡張セットを使用する場合は、それらを設定ファイルに保存すると便利です。

@data オプションを使用すると、コマンド・ラインで環境変数と設定ファイルを指定できます。設定ファイルを指定するには、***data*** を設定ファイルのパスと名前で置き換えます。同じ名前の環境変数と設定ファイルが存在する場合は、環境変数が使用されます。

設定ファイルには、改行を含めたり、@data オプションを含むあらゆるオプションの設定を格納したりできます。数値記号 (#) を使用すると、行をコメントとして指定することができます。

@data パラメータはコマンド・ラインの任意の位置に指定でき、ファイルに含まれるパラメータがその位置に挿入されます。さらに @data は、1 つのコマンド・ラインで複数回使用して複数の設定ファイルを指定できます。

ユーティリティは、指定の設定ファイルを展開し、コマンド・ライン全体を左から右へ読み取ります。コマンド・ライン内のその他のオプションで上書きされるオプションを指定した場合、行の終端に近い方のオプションが有効になります。場合によっては、競合するオプションのためにエラーが発生することがあります。

注意
サーバ・バックグラウンド起動ユーティリティ (dbspawn) では、@data オプションで指定された設定ファイルは展開されません。

設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。

設定ファイルの内容の難読化については、「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。

例

次の設定ファイルには、検証ユーティリティ (dbvalid) の一連のオプションが含まれています。

```
#Connect to the sample database as the user DBA with password sql
-c "UID=DBA;PWD=sql;DBF=samples-dir¥demo.db"
#Perform an express check on each table
-fx
#Log output messages to the specified file
-o "c:¥validationlog.txt
```

samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

この設定ファイルを *c:¥config.txt* として保存すると、コマンドで次のように使用できます。

```
dbvalid @c:¥config.txt
```

設定ファイルでの条件付き解析の使用

設定ファイルで条件付き解析を使用することによって、そのファイルを使用できるユーティリティを指定できます。条件ディレクティブを使用すると、設定ファイルを使用するユーティリティに応じてコマンド・パラメータを適用したり除外したりすることができます。設定ファイルで条件付き解析を使用する場合でも、ファイル非表示ユーティリティ (dbfhide) を使用して、設定ファイルの内容を非表示にすることができます。

構文

configuration-file= text...

text : *comment* | *conditional* | *command-line-option*

comment : *line starting with # that is not a conditional*

conditional :

#if *condition*
text
[**#elif** *condition*
text
] ...
[**#else**
text
] ...
#endif

condition : { **tool=***utility-name*[,*utility-name*]... | *utility-name* }

utility-name では、次の値がサポートされています。

dbbackup	dbinfo	dbltm	dbstop	dbxtract
dbdsn	dbinit	dbmlsync	dbsupport	mlsrv10

dbeng10	dblic	dbping	dbsvc	mluser
dberase	dblocate	dbremote	dbunload	qaagent
dbfhide	dblog	dbspawn	dbupgrad	rteng10
dbhist	dblsn	dbsrv10	dbvalid	

使用法

ディレクティブと認識されるようにするには、1 つの行で最初の空白以外の文字を # とします。#if または #elif ディレクティブ内にユーティリティが含まれている場合、そのディレクティブから次の条件付きディレクティブまでの行が処理の対象となります。#else ディレクティブは、それより前のブロックにユーティリティがない場合に実行する処理を示します。条件付きディレクティブ構造は、#endif ディレクティブで終了します。

tool= で指定するツール名のリスト内に空白を入れることはできません。条件付きディレクティブはネストすることができます。設定ファイルの解析中にエラーが発生すると、ユーティリティは設定ファイルを開くことができない旨をレポートします。

例

次の設定ファイルを使用できるユーティリティは、dbping、dbstop、dbvalid です。

```
#if tool=dbping,dbstop,dbvalid
  #always make tools quiet
  -q
  -c "UID=DBA;PWD=sql;ENG=myserver;DBN=mydb"
  #if dbping
    #make a database connection
    -d
  #elif tool=dbstop
    #don't ask
    -y
  #else
    #must be dbvalid
    #use WITH EXPRESS CHECK
    -fx
  #endif
#endif
```

バックアップ・ユーティリティ (dbbackup)

実行中のデータベースのデータベース・ファイルやトランザクション・ログについて、クライアント側のバックアップを作成します。

構文

dbbackup [*options*] *target-directory*

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-b *block-size*	このオプションでは、データベースから dbbackup へのページ転送に使用するブロックの最大サイズ (ページ数) を指定します。dbbackup ユーティリティは、指定された数のページを割り付けようとします。割り付けが失敗した場合は、この値を半分にして再試行し、成功するまでこれを繰り返します。デフォルトのサイズは 128 です。
-c *"keyword=value; ⋯"*	接続パラメータを指定します。データベースに接続するためには、DBA 権限または REMOTE DBA 権限 (SQL Remote) のあるユーザ ID を使用する必要があります。「接続パラメータ」 230 ページを参照してください。 たとえば、次のコマンドは、DBA ユーザとして サーバ sample_server に接続し、このサーバ上のサンプル・データベースを *SQLAnybackup* ディレクトリにバックアップします。 `dbbackup -c "ENG=sample_server;DBN=demo;UID=DBA;PWD=sql" SQLAnybackup`
-d	トランザクション・ログ・ファイルがあっても、それをバックアップしないで、メイン・データベース・ファイルだけをバックアップします。

オプション	説明
-k *checkpoint-log-copy-option*	このオプションでは、バックアップ先のディレクトリに書き込む前に dbbackup でデータベース・ファイルに対してどのような処理を実行するかを指定します。バックアップ中に更新前イメージを適用するか、チェックポイント・ログをバックアップとしてコピーするかを選択します。どちらを選択するかによって、パフォーマンスに違いが生じます。-s オプションを指定してサーバ側でバックアップを実行する場合、-k のデフォルト設定は auto です。それ以外の場合は、copy がデフォルト設定となります。 ♦ **auto** auto を指定すると、データベース・サーバはバックアップ・ディレクトリがあるボリュームの空きディスク領域サイズをチェックします。バックアップを開始する時点でデータベース・サイズの 2 倍以上の空きディスク領域がある場合は、copy を指定した場合と同じ方法でバックアップが実行されます。それ以外の場合は、nocopy を指定した場合と同じ方法でバックアップが実行されます。この設定は、-s を指定した場合のみ使用できます。 ♦ **copy** copy を指定すると、バックアップ中には、変更されたページの更新前イメージは適用されません。チェックポイント・ログの全体とシステム DB 領域がバックアップ・ディレクトリにコピーされます。このデータベースを次に起動すると、データベースは自動的にバックアップ開始時のチェックポイントまでリカバリされます。 このオプションを使用すると、ページの更新前イメージをテンポラリ・ファイルに書き込む必要がないため、バックアップ・パフォーマンスが向上し、バックアップ中に動作中の他の接続との内部サーバ競合が減少します。ただし、データベース・ファイルのバックアップ・コピーにはチェックポイント・ログが含まれ、このログにはバックアップの開始以降に変更されたページの更新前イメージが保存されているため、バックアップ・コピーのサイズがバックアップを開始した時点のデータベース・ファイルより大きくなることがあります。copy オプションは、バックアップ先ディレクトリのディスク領域が十分である場合に使用してください。 ♦ **nocopy** nocopy を指定すると、チェックポイント・ログがバックアップとしてコピーされません。この場合、変更されたページの更新前イメージはテンポラリ・ファイルに保存され、バックアップの実行中、バックアップに適用されます。データベース・ファイルのバックアップ・コピーは、バックアップを開始した時点のデータベースと同じサイズになります。バックアップ・コピーは、チェックポイント・ログが含まれていないため、実際には多少小さくなることがありますこのオプションを指定すると、データベース・ファイルのバックアップは小さくなりますが、バックアップの速度が遅くなり、データベース・サーバで実行される他の操作のパフォーマンスが低下することがあります。バックアップ先のドライブの空き領域が少ない場合に、このオプションが役に立ちます。 ♦ **recover** recover を指定すると、(copy オプションを指定した場合と同じように) チェックポイント・ログがコピーされますが、このチェックポイント・ログはバックアップの完了時にデータベースに適用されます。したがって、データベース・ファイルのバックアップは、バックアップ操作を開始した時点と同じ状態 (および同じサイズ) となります。このオプションが役に立つのは、バックアップ・ドライブの空き領域が少ない場合です (チェック

オプション	説明
-l *filename*	このオプションは、サーバがクラッシュした場合に第 2 のシステムをすばやく起動するために用意されています。サーバが実行されている間、ライブ・バックアップは終了することがなく実行を続けます。ライブ・バックアップは、プライマリ・サーバが使用できなくなるまで実行されます。クラッシュが発生した時点でライブ・バックアップは停止しますが、バックアップされたログ・ファイルはそのまま残り、第 2 のシステムを即座に起動するために使用できます。「ライブ・バックアップとトランザクション・ログ・ミラーの違い」 832 ページと「ライブ・バックアップの作成」 856 ページを参照してください。 -l を指定する場合は、-s を使用してサーバ上にバックアップを作成できません。
-n	このオプションは、-r と共に使用するもので、バックアップ・トランザクション・ログ・ファイルの命名規則を *yymmddxx.log* に変更します。ここで、*xx* は **AA** から **ZZ** までの連続した英字を表し、*yymmdd* は現在の年月日を表します。 トランザクション・ログ・ファイルのバックアップ・コピーは、*yymmddxx.log* の命名規則に基づいて、コマンドで指定したディレクトリに保存されます。これにより、トランザクション・ログ・ファイルの複数のバージョンのバックアップを、同じバックアップ・ディレクトリに保存することができます。 また、-x オプションと -n オプションの両方を使用して、ログ・コピーの名前を変更することもできます。例を示します。 `dbbackup -c "UID=DBA;PWD=sql" -x -n mybackupdir`
-o *filename*	指定したファイルに、出力メッセージを書き込みます。
-q	出力メッセージを表示しません。このオプションは、このユーティリティをコマンド・プロンプトから実行する場合のみ使用できます。

オプション	説明
-r	このオプションは、トランザクション・ログの名前を変更して新しいログを開始します。このオプションを使用すると、チェックポイントが発生し、次の 3 つの手順が発生します。 1. 現在作業しているトランザクション・ログ・ファイルのコピーが作成され、コマンドで指定したディレクトリに保存されます。 2. 現在のトランザクション・ログは現在のディレクトリ内に残りますが、フォーマット *yymmddxx.log* を使用して名前が変更されます。ここで、*xx* は *AA* で始まる *ZZ* までの連続した英字を表し、*yymmdd* は現在の年月日を表します。このファイルは、現在のトランザクション・ログではなくなります。 3. トランザクションを含まない新しいトランザクション・ログ・ファイルが作成されます。このファイルに、直前まで現在のトランザクション・ログとして使用されていたファイルの名前が付けられ、データベース・サーバによって現在のトランザクション・ログとして使用されます。
-s	このオプションにより、BACKUP DATABASE 文を使用して、サーバ上にイメージのバックアップを作成できます。-s オプションを指定する場合、-l オプション (トランザクション・ログのライブ・バックアップを作成) は使用できません。指定されたディレクトリは、サーバの現在のディレクトリに対する相対パスなので、完全パス名を指定することをおすすめます。さらに、サーバには指定されたディレクトリへの書き込みパーミッションが必要です。-s を指定すると、バックアップ・ユーティリティは進行メッセージを表示せず、既存のファイルを上書きするときにプロンプトを表示しません。既存のファイルを上書きするときにプロンプトを表示させる場合は、-s または -y を指定しないでください。-k リカバリ・オプションを指定する場合は、-s オプションも必ず指定してください。
-t	トランザクション・ログをデータベース・ファイルの最新のバックアップ・コピーに対して適用できるので、このオプションはインクリメンタル・バックアップとして使用できます。
-x	既存のトランザクション・ログをバックアップし、次に元のログを削除して、新しいトランザクション・ログを起動します。
-xo	現在のトランザクション・ログを削除して、新しいトランザクション・ログを起動します。この操作では、バックアップは実行されません。この操作の目的は、レプリケーション環境以外の環境でディスク領域を解放することです。

オプション	説明
-y	このオプションを指定すると、確認メッセージを表示することなく、バックアップ・ディレクトリが作成されるか、ディレクトリ内の既存のバックアップ・ファイルが置き換えられます。既存のファイルを上書きするときにプロンプトを表示させる場合は、-s または -y を指定しないでください。
target-directory	バックアップ・ファイルのコピー先ディレクトリ。このディレクトリが存在しない場合は作成されます。ただし、親ディレクトリが存在していなければなりません。

備考

バックアップ・ユーティリティを使うと、すべてのバックアップ・コピーを単一データベースに作成することができます。単純なデータベースは、メイン・データベース・ファイルとトランザクション・ログの 2 つのファイルからなります。より複雑なデータベースは、複数のファイルにテーブルを格納できます。各ファイルは個別の DB 領域となります。すべてのバックアップ・データベース・ファイル名はデータベース・ファイル名と同じです。バックアップ・ユーティリティで作成したイメージのバックアップは、バックアップされた各ファイルの別ファイルで構成されます。

アーカイブ・バックアップ (データベース・ファイルとトランザクション・ログが 1 つのファイルに保存される) 作成の詳細については、「アーカイブ・バックアップの作成」 826 ページを参照してください。

実行中のデータベースに対してバックアップ・ユーティリティを使用するのは、データベースが実行されていないときにデータベース・ファイルをコピーするのと同じです。バックアップ・ユーティリティを使用すると、他のアプリケーションやユーザーがデータベースを使用しているときでも、そのデータベースをバックアップできます。

-d または -t のいずれのオプションも使用されていない場合は、すべてのデータベース・ファイルがバックアップされます。

バックアップ・ユーティリティは、データベース・ファイルのクライアント側のバックアップを作成します。BACKUP DATABASE 文に -s オプションを指定すると、サーバ上にバックアップを作成できます。

サーバ側でのバックアップ実行の詳細については、「BACKUP 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

dbbackup のほかに、次の方法を使用して、バックアップ・ユーティリティにアクセスできます。

- Sybase Central の [バックアップ・イメージ作成] ウィザードを使用する。「イメージ・バックアップの作成」 821 ページを参照してください。
- Interactive SQL の BACKUP DATABASE 文を使用する。「BACKUP 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

推奨されるバックアップ・プロシージャの詳細については、「バックアップとデータ・リカバリ」 811 ページを参照してください。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

終了コードの詳細については、「ソフトウェア・コンポーネントの終了コード」『SQL Anywhere サーバ - プログラミング』を参照してください。

データ・ソース・ユーティリティ (dbdsn)

SQL Anywhere ODBC データ・ソースの作成、削除、説明、およびリスト作成を行います。

構文

dbdsn [*modifier-options*]
{ **-l**[**s** | **u**] [**-qq**]
| **-d**[**s** | **u**] *dsn*
| **-g**[**s** | **u**] *dsn*
| **-w**[**s** | **u**] *dsn* [*details-options*;⋯]
| **-cl**[**-qq**] }

オプション

主要オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-l[**s** | **u**] [**qq**]	使用可能な SQL Anywhere ODBC データ・ソースをリストします。リストの形式は -b オプションまたは -v オプションを使用して変更できます。Windows では、**u** (ユーザ) または **s** (システム) 指定子を使用して、オプションを修正できます。デフォルトの指定子は **u** です。
-d[**s** | **u**] *dsn*	指定の SQL Anywhere データ・ソースを削除します。-y を指定すると、確認メッセージを表示せずに既存のデータ・ソースが削除されます。Windows では、**u** (ユーザ) または **s** (システム) 指定子を使用して、オプションを修正できます。デフォルトの指定子は **u** です。
-g[**s** | **u**] *dsn*	指定された SQL Anywhere データ・ソースの定義をリストします。-b オプションまたは -v オプションを使用して、出力のフォーマットを修正できます。Windows では、**u** (ユーザ) または **s** (システム) 指定子を使用して、オプションを修正できます。デフォルトの指定子は **u** です。
-w[**s** | **u**] *dsn* [*details-options*]	新しいデータ・ソースを作成します。同じ名前のデータ・ソースが存在する場合は上書きします。-y を指定すると、確認メッセージを表示せずに既存のデータ・ソースを上書きします。Windows では、**u** (ユーザ) または **s** (システム) 指定子を使用して、オプションを修正できます。デフォルトの指定子は **u** です。

主要オプション	説明
-cl[**-qq**]	この便利なオプションは、dbdsn ユーティリティがサポートしている接続パラメータをリストします。このオプションと一緒に -qq を使用すると、メッセージまたはタイトルなしで、利用可能な接続パラメータがリストされます。「接続パラメータ」 230 ページを参照してください。 サポートされている ODBC 接続パラメータの詳細については、「ODBC 接続パラメータ」 650 ページを参照してください。

変更オプション	説明
-b	リストの出力を単一の行の接続文字列にフォーマットします。
-cm	データ・ソースの作成に使用するコマンドを表示します。このオプションを使用すると、作成コマンドをファイルに出力できます。作成コマンドは別のコンピュータにデータ・ソースを追加したり、変更が加えられたデータ・ソースを元の状態にリストアするために使用できます。-cm とともに -g オプションまたは -l オプションを指定しないとコマンドは失敗します。-g を指定すると、指定のデータ・ソースの作成コマンドが表示されるのに対し、-l を指定するとすべてのデータ・ソースの作成コマンドが表示されます。 指定のデータ・ソースが存在しない場合は、データ・ソースを削除するコマンドが生成されます。たとえば、コンピュータに mydsn データ・ソースが存在しない場合、dbdsn -cm -g mydsn は次のコマンドを返して mydsn データ・ソースを削除します。 `dbdsn -y -du "mydsn"`
-dr	DSN を表示するときに Driver パラメータを含めます。このオプションが特に役立つのは、-cm オプションを使用して DSN を再作成する場合です。このオプションによって、dbdsn の現バージョンで異なるバージョンの ODBC ドライバを参照する DSN を作成することができます。 たとえば、次のようなコマンドを使用してバージョン 9.0 の DSN を作成したとします。 `dbdsn -y -wu "9.0 Student Sample" -c "UID=DBA;PWD=sql;...;Driver= Adaptive Server Anywhere 9.0"` dbdsn -cm -l を実行すると、同じコマンドが Driver= パラメータなしで表示されます。これによって、SQL Anywhere バージョン 10.0 の ODBC ドライバを使用して DSN が再作成されます。 それに対し、dbdsn -dr -cm -l を実行すると、Driver= パラメータが含められるため、バージョン 9 の ODBC ドライバを使用した場合とまったく同じデータ・ソースが再作成されます。

変更オプション	説明
-f	このオプションを指定すると、使用されるシステム・ファイルの名前が表示されます。このオプションは UNIX でのみ使用できます。
-ns	UNIX 上でデータ・ソースを作成するときに -ns オプションを使用すると、環境変数の設定を使用してシステム情報ファイル (デフォルトのファイル名は *.odbc.ini*) のロケーションを確認するように指定できます。このオプションは、使用される可能性のあるシステム情報ファイルが複数ある場合に、dbdsn が使用するファイルを特定する目的にも利用できます。 データ・ソースを作成するときに -ns オプションを指定しなかった場合、システム情報ファイルはユーザのホーム・ディレクトリとそのパス内で検索されます。 システム情報ファイルがどのように検索されるかについては、「UNIX での ODBC データ・ソースの使用」 81 ページを参照してください。
-o *filename*	指定したファイルに、出力メッセージを書き込みます。
-or	-c オプションとともに指定すると、iAnywhere Solutions Oracle ドライバのデータ・ソースが作成されます。次に例を示します。 `dbdsn -w MyOracleDSN -or -c Userid=DBA;Password=sql;ServerName=abcd;ArraySize=500;ProcResults=y` -cl オプションを -or オプションとともに指定すると、iAnywhere Solutions Oracle ドライバの接続パラメータのリストを取得できます。 詳細については、「iAnywhere Solutions Oracle ドライバ」『Mobile Link - サーバ管理』を参照してください。
-pe	このオプションを指定すると、DSN に PWD エントリがある場合、PWD エントリ内のパスワードが暗号化され、PWD エントリは暗号化されたパスワードが設定された ENP エントリに置き換えられます。
-q	サーバ・メッセージ・ウィンドウへの出力を抑制します。データ・ソースの削除時または変更時に -q を指定する場合は、-y も指定してください。
-qq	サーバ・メッセージ・ウィンドウへの出力を抑制します。-o オプションを指定すると、タイトルも表示されません。このオプションは、-l オプションと -cl オプションが指定された場合のみに使用できます。
-v	数行のリスト出力を表としてフォーマットします。

変更オプション	説明
-y	確認を要求するプロンプトを表示しないで、自動的に各データ・ソースを削除または上書きします。データ・ソースの削除時または変更時に -q を指定する場合は、-y も指定してください。

詳細オプション	説明
-c "*keyword*=*value*;⋯"	接続パラメータを接続文字列として指定します。「接続パラメータ」 230 ページを参照してください。
-cw	(-c で指定された) DBF パラメータが確実に絶対ファイル名になるようにします。DBF の値が絶対ファイル名でない場合、dbdsn は現在の作業ディレクトリ (CWD) を付加します。オペレーティング・システムの中には、バッチ・ファイルですぐに利用できる CWD 情報がないものもあるので、このオプションは便利です。

備考

データ・ソース・ユーティリティは、プラットフォームを問わないユーティリティで、ODBC アドミニストレータに代わり SQL Anywhere ODBC データ・ソースの作成、削除、記述、リストを実行します。このユーティリティは、バッチ処理に役立ちます。Windows オペレーティング・システムでは、データ・ソースはレジストリに保存されます。

ODBC アドミニストレータを使用した Windows でのデータ・ソース作成の詳細については、「ODBC データ・ソースの使用」 75 ページを参照してください。

UNIX オペレーティング・システムでは、データ・ソースはシステム情報ファイル (デフォルトのファイル名は *.odbc.ini*) に保持されます。データ・ソース・ユーティリティを使用して SQL Anywhere ODBC データ・ソースを UNIX 上で作成または削除すると、システム情報ファイルの [ODBC データ・ソース] セクションが自動的に更新されます。UNIX で -c オプションを使用して Driver 接続パラメータを指定しない場合、データ・ソース・ユーティリティによって SQLANY10 環境変数の設定に基づき SQL Anywhere ODBC ドライバのフル・パスを使って Driver エントリが自動的に追加されます。

システム情報ファイルの詳細については、「UNIX での ODBC データ・ソースの使用」 81 ページを参照してください。

警告
SQL Anywhere データ・ソースのみを使用する場合以外は、UNIX でファイル非表示ユーティリティ (dbfhide) を使って *.odbc.ini* システム情報ファイルを難読化しないでください。他のデータ・ソース (Mobile Link 同期など) を使用する予定の場合、システム情報ファイルを難読化すると、他のドライバが正しく機能しなくなることがあります。

変更オプションは、主要オプションの前または後に指定できます。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」 『SQL Anywhere サーバ - プログラミング』を参照してください。

ODBC 接続パラメータ

データ・ソース・ユーティリティ (dbdsn) は以下の ODBC 接続パラメータをサポートしています。ブール (true または false) 引数は、true の場合は YES または 1、false の場合は NO または 0 のいずれかです。

名前	説明
Delphi	Delphi では、1 ローにつき複数のブックマーク値を処理できません。この値を NO に設定すると、各ローに 1 つ (YES の場合は 2 つ) のブックマーク値が割り当てられます。このオプションを YES に設定すると、スクロール可能なカーソルのパフォーマンスが向上します。
DescribeCursor	このパラメータにより、プロシージャが実行されたときにカーソルを再記述する頻度を指定できます。デフォルト設定は [要求に応じて] です。 ◆ **[しない]** カーソルを再記述する必要がない場合には、このオプションを 0、N、または NO に指定します。カーソルの再記述は負荷が高く、パフォーマンスを低下させる可能性があります。 ◆ **[要求に応じて]** カーソルを再記述する必要があるかどうかを ODBC ドライバが決定するようにするには、1、Y、または YES を指定します。プロシージャに RESULT 句があると、ODBC アプリケーションは、カーソルを開いた後結果セットを再記述できません。これはデフォルト設定です。 ◆ **[常に]** 2、A、または ALWAYS を指定すると、カーソルを開くたびに再記述します。Transact-SQL プロシージャや、複数の結果セットを返すプロシージャを使用する場合は、カーソルを開くたびに再記述する必要があります。
Description	このパラメータにより、ODBC データ・ソースの説明を入力できます。
Driver	このパラメータでは、接続の ODBC ドライバを Driver=<*driver-name*> の形式で指定できます。デフォルトで使用されるドライバは **SQL Anywhere 10** です。*driver-name* は、SQL Anywhere *X* にしてください。その場合、*X* にはソフトウェアのメジャー・バージョン番号を指定します。*driver-name* が **SQL Anywhere** で始まらない場合、データ・ソース・ユーティリティ (dbdsn) はそれを読み取ることができません。 UNIX では、このパラメータによって共有オブジェクトへの完全に修飾されたパスが指定されます。UNIX で Driver 接続パラメータを指定しない場合、データ・ソース・ユーティリティによって SQLANY10 環境変数の設定に基づき SQL Anywhere ODBC ドライバのフル・パスを使って Driver エントリが自動的に追加されます。
GetTypeInfoChar	このオプションを YES に設定すると、CHAR カラムは SQL_VARCHAR ではなく SQL_CHAR として返されます。デフォルトでは、CHAR カラムは SQL_VARCHAR として返されます。

名前	説明
InitString	InitString により、接続確立後すぐに実行されるコマンドを指定できます。たとえば、データベース・オプションを設定したり、ストアド・プロシージャを実行したりできます。
IsolationLevel	以下の値の 1 つを指定して、データ・ソースの初期独立性レベルを設定できます。 ◆ **0** これはコミットされない読み込み独立性レベルとも呼ばれます。これはデフォルトの独立性レベルです。これは最大レベルの同時実行性を提供しますが、結果セットにダーティ・リード、繰り返し不可能読み出し、幻ローが見受けられる場合があります。 ◆ **1** これはコミットされた読み込みレベルとも呼ばれます。レベル 0 よりも低い同時実行性を提供しますが、レベル 0 の結果セットに見られる不整合性が一部解消されます。繰り返し不可能読み出しや幻ローが発生することはありますが、ダーティ・リードは発生しません。 ◆ **2** これは繰り返し読み出しレベルとも呼ばれます。幻ローが発生することがあります。ダーティ・リードと繰り返し不可能ローは発生しません。 ◆ **3** これは直列化可能レベルとも呼ばれます。これは最低レベルの同時実行性を提供する、最も厳しい独立性レベルです。ダーティ・リード、繰り返し不可能読み出し、幻ローは発生しません。 ◆ **snapshot** この独立性レベルを使用するには、データベースのスナップショット・アイソレーションを有効にする必要があります。スナップショット・アイソレーションのレベルは、読み込みと書き込み間の干渉を防ぎます。書き込みは相互に干渉する可能性があります。一貫性のない動作が多少生じる可能性がありますが、パフォーマンスは競合に関して独立性レベルを 0 に設定した場合と同じです。 ◆ **statement-snapshot** この独立性レベルを使用するには、データベースのスナップショット・アイソレーションを有効にする必要があります。スナップショット・アイソレーションのレベルは、読み込みと書き込み間の干渉を防ぎます。書き込みは相互に干渉する可能性があります。一貫性のない動作が多少生じる可能性がありますが、パフォーマンスは競合に関して独立性レベルを 0 に設定した場合と同じです。 ◆ **readonly-statement-snapshot** これは、独立性レベルとも呼ばれます。この独立性レベルを使用するには、データベースのスナップショット・アイソレーションを有効にする必要があります。スナップショット・アイソレーションのレベルは、読み込みと書き込み間の干渉を防ぎます。書き込みは相互に干渉する可能性があります。一貫性のない動作が多少生じる可能性がありますが、パフォーマンスは競合に関して独立性レベルを 0 に設定した場合と同じです。 詳細については、「独立性レベルの選択」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

名前	説明
KeysInSQLStatistics	YES を指定すると、SQLStatistics 関数から外部キーが返されるようになります。ODBC 仕様では、SQLStatistics によってプライマリ・キーと外部キーが戻されないように指定しています。しかし、一部の Microsoft アプリケーション (Visual Basic や Access など) では、SQLStatistics によってプライマリ・キーと外部キーが戻されることを前提にしています。
LazyAutocommit	文が完了するまでコミット操作を遅延させるには、このパラメータを YES に設定します。
PrefetchOnOpen	PrefetchOnOpen が YES に設定されていると、カーソルを開く要求を含むプリフェッチ要求が送信されます。プリフェッチにより、カーソルを開くたびにネットワーク要求が行われることはなくなります。カーソルを開くときにプリフェッチを実行するには、カラムをバインドしておきます。この接続パラメータにより、クライアント／サーバ要求の数が削減され、LAN や WAN を介したパフォーマンスが向上します。
PreventNotCapable	SQL Anywhere ODBC ドライバは、修飾子をサポートしていないため「**ドライバが動作しません。**」というエラーを返します。ODBC アプリケーションの中には、このエラーを適切に処理しないものもあります。このようなアプリケーションでも作業できるように、このエラー・コードが返されないようにするには、このパラメータを YES に設定します。
SuppressWarnings	フェッチ時にデータベース・サーバから返される警告メッセージを表示しない場合は、このパラメータを YES に設定します。バージョン 8.0.0 以降のデータベース・サーバでは、それよりも前のバージョンのソフトウェアに比べて多様なフェッチ警告が返されます。以前のバージョンのソフトウェアを使用して配備されたアプリケーションに対して、フェッチの警告を適切に処理するためにこのオプションを選択できます。
TranslationDLL	このオプションは下位互換性のために提供されています。トランスレータの使用はおすすめできません。
TranslationName	このオプションは下位互換性のために提供されています。トランスレータの使用はおすすめできません。
TranslationOption	このオプションは下位互換性のために提供されています。トランスレータの使用はおすすめできません。

参照

- 「ODBC データ・ソースの使用」 75 ページ
- 「UNIX での ODBC データ・ソースの使用」 81 ページ

例

データ・ソース newdsn の定義を書き込みます。データ・ソースがすでに存在する場合でも、確認メッセージは表示しません。

```
dbdsn -y -w newdsn -c "UID=DBA;PWD=sql;LINKS=TCPIP;ENG=myserver"
```

次のように、オプションを別の順序で指定することもできます。

```
dbdsn -w newdsn -c "UID=DBA;PWD=sql;LINKS=TCPIP;ENG=myserver" -y
```

既知のすべてのユーザ・データ・ソースをリストします (1 行に 1 つのデータ・ソース名)。

```
dbdsn -l
```

既知のすべてのシステム・データ・ソースをリストします (1 行に 1 つのデータ・ソース名)。

```
dbdsn -ls
```

すべてのデータ・ソースを関連する接続文字列とともにリストします。

```
dbdsn -l -b
```

ユーザ・データ・ソース MyDSN 用の接続文字列をレポートします。

```
dbdsn -g MyDSN
```

システム・データ・ソース MyDSN 用の接続文字列をレポートします。

```
dbdsn -gs MyDSN
```

最初に BadDSN の接続パラメータをリストし、確認メッセージを表示してから、データ・ソース BadDSN を削除します。

```
dbdsn -d BadDSN -v
```

確認メッセージを表示せずに、データ・ソース BadDSN を削除します。

```
dbdsn -d BadDSN -y
```

データベース・サーバ MyServer のデータ・ソース NewDSN を作成します。

```
dbdsn -w NewDSN -c "UID=DBA;PWD=sql;ENG=MyServer"
```

NewDSN がすでに存在する場合は、データ・ソースを上書きするかどうか確認を求められます。

すべての接続パラメータ名とそのエイリアスをリストします。

```
dbdsn -cl
```

すべてのユーザ DSN をリストします (メッセージまたはタイトルなし)。

```
dbdsn -l -qq -o dsninfo.txt
```

すべての接続パラメータをリストします (メッセージまたはタイトルなし)。

```
dbdsn -cl -qq -o dsninfo.txt
```

絶対ファイル名を指定します。DSN が作成されている場合、*DBF=c:¥SQLAnywhere10¥my.db* が含まれます。

```
c:¥SQLAnywhere10> dbdsn -w testdsn -cw -c UID=DBA;PWD=sql;ENG=SQLAny;DBF=my.db
```

SQL Anywhere 10 デモ・データ・ソースを作成して、*restoredsn.bat* というファイルに出力するコマンドを生成します。


```
dbdsn -cm -g "SQL Anywhere 10 Demo" > restoredsn.bat
```

restoredsn.bat ファイルには以下が含まれています。

```
dbdsn -y -wu "SQL Anywhere 10 Demo" -c "UID=DBA;PWD=sql;
DBF='C:¥Documents and Settings¥All Users¥Documents¥SQL Anywhere 10¥Samples¥demo.db';
ENG=demo10;START='C:¥Program Files¥SQL Anywhere 10¥win32¥dbeng10.exe';
ASTOP=yes;Description='SQL Anywhere 10 Sample Database'"
```

次のコマンドは、UNIX のシステム情報ファイルのロケーションを返します。

```
dbdsn -f
```

このコマンドは次の出力を返します。

```
dbdsn using /home/user/.odbc.ini
```

次のコマンドは、システム情報ファイルのロケーションを変更します。

```
export ODBCINI=./myodbc.ini
```

次のコマンドは、dbdsn -f を使用してシステム情報ファイルの新しいロケーションを確認します。

```
dbdsn using ./myodbc.ini
```

次のコマンドは、-ns オプションを使用してデータ・ソースを作成します。

```
dbdsn -w NewDSN -c "UID=DBa" -ns
```

このコマンドは次の出力を返します。

```
Configuration "newdsn" written to file ./myodbc.ini
```

消去ユーティリティ (dberase)

データベースに関連する DB 領域とトランザクション・ログ・ファイルを消去します。

構文

dberase [*options*] *database-file*

オプション	説明
@data	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-ek *key*	このオプションを使用すると、強力に暗号化されているデータベースの暗号化キーをコマンドに直接指定できます。データベースが強力に暗号化されている場合、データベースまたはトランザクション・ログを使用するには必ず暗号化キーを指定します。強力な暗号化が適用されたデータベースの場合、-ek または -ep のどちらかを指定します。両方同時には指定できません。強力に暗号化されたデータベースでは、正しいキーを指定しないとコマンドが失敗します。
-ep	このオプションを使用すると、暗号化キーの入力を求めるプロンプトを表示するよう指定できます。このオプションでは、暗号化キーを入力するためのダイアログ・ボックスが表示されます。クリア・テキストでは暗号化キーを見ることができないようにすることで、高いセキュリティが得られます。強力な暗号化が適用されたデータベースの場合、-ek または -ep のどちらかを指定します。両方同時には指定できません。強力に暗号化されたデータベースでは、正しいキーを指定しないとコマンドが失敗します。
-o *filename*	指定したファイルに、出力メッセージを書き込みます。
-q	クワイエット・モードで実行し、出力メッセージを表示しません。このオプションを指定する場合、-y も指定しないと操作は失敗します。
-y	このオプションを指定すると、確認メッセージを表示することなく、各ファイルを削除します。-q を指定する場合、-y も指定しないと操作は失敗します。

備考

消去ユーティリティを使って、データベース・ファイルと、それに関連するトランザクション・ログを消去できます。または、トランザクション・ログ・ファイルやトランザクション・ログ・ミラー・ファイルを消去できます。すべてのデータベース・ファイルとトランザクション・ログ・ファイルに読み込み専用のマークを付けて、データベースが突然損傷を受けたり、データベース・ファイルが不用意に削除されないようにします。

database-file は、データベース・ファイルまたはトランザクション・ログ・ファイルです。ファイル名は、拡張子も含めてすべて指定してください。データベース・ファイルを指定すると、関連するトランザクション・ログ・ファイルが (ミラーもある場合はそれも含めて) 消去されます。

注意
消去ユーティリティは DB 領域を*消去しません*。DB 領域を消去したい場合、DROP DATABASE 文を使用するか、または Sybase Central の [データベース消去] ウィザードを使用します。「DROP 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
[データベース消去] ウィザードでは、DB 領域やトランザクション・ログ・ファイルを消去することもできます。「データベースの消去」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

他の DB 領域を参照するデータベース・ファイルを削除しても、DB 領域ファイルが自動的に削除されることはありません。DB 領域ファイルを手動で削除したい場合は、ファイルを読み込み専用から書き込み可能に変更し、次にファイルを 1 つずつ削除します。別の方法として、DROP DATABASE 文を使用して、データベースと関連する DB 領域ファイルを消去することもできます。

データベース・ファイルを消去すると、関連するトランザクション・ログとトランザクション・ミラーも削除されます。トランザクション・ログ・ミラーも保有しているデータベースのトランザクション・ログを消去しても、ミラーは削除されません。

このユーティリティの使用中に、消去するデータベースを起動しないでください。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」『SQL Anywhere サーバ - プログラミング』を参照してください。

ファイル非表示ユーティリティ (dbfhide)

単純暗号化を使用して、設定ファイルと初期化ファイルの内容を非表示にします。

構文

dbfhide *original-configuration-file encrypted-configuration-file*

オプション	説明
original-configuration-file	元のファイルの名前を指定します。
encrypted-configuration-file	難読化された新しいファイルの名前を指定します。

備考

一部のユーティリティでは、コマンド・ライン・オプションを保存するために設定ファイルが使用されます。これらのオプションにパスワードを含めることができます。ファイル非表示ユーティリティを使用して、設定ファイル、および SQL Anywhere とそのユーティリティで使用する *.ini* ファイルに単純暗号化を追加することによって、ファイルの内容を難読化できます。元のファイルは変更されません。一度ファイルに追加した単純暗号化を削除することはできません。難読化されたファイルに変更を加えるためには、再度変更したり難読化したりできるように元のファイルのコピーを保存しておく必要があります。

設定ファイルの使用については、「設定ファイルを使用したサーバ起動オプションの保存」 19 ページを参照してください。

暗号化の詳細については、「安全なデータの管理」 917 ページを参照してください。

.ini ファイルの内容の非表示

多くの場合、SQL Anywhere では *.ini* ファイルに特定の名前が付けられていると予測します。名前が重要なファイル (*saldap.ini* など) に単純暗号化を追加する場合、元のファイルのコピーを別の名前を使って保存してください。元のファイルのコピーを保存していない場合、ファイルがいったん難読化されると、その内容を変更できません。次の手順では、*.ini* ファイルに単純暗号化を追加する方法について説明します。

♦ **ファイルの内容を非表示にするには、次の手順に従います。**

1. ファイルを別の名前で保存します。

   ```
   rename saldap.ini saldap.ini.org
   ```

2. ファイル非表示ユーティリティを使用してファイルを難読化し、難読化されたファイルに必要なファイル名を付けます。

   ```
   dbfhide saldap.ini.org saldap.ini
   ```

3. ファイル・システムまたはオペレーティング・システムの保護により *saldap.ini.org* ファイルを保護するか、安全な場所に保存します。

   saldap.ini ファイルに変更を加えるには、*saldap.ini.org* ファイルを編集し、手順 2 を繰り返します。

警告
SQL Anywhere データ・ソースだけを使用している場合を除き、UNIX 上でファイル非表示ユーティリティ (dbfhide) を使用して、システム情報ファイル (デフォルトのファイル名は *.odbc.ini*) に単純暗号化を追加しないでください。他のデータ・ソース (Mobile Link 同期など) を使用する予定の場合、システム情報ファイルの内容を難読化すると、他のドライバが正しく機能しなくなることがあります。

このユーティリティは、設定ファイルからオプションを読み込む ***@data*** *パラメータを受け入れません。*

例

パーソナル・データベース・サーバとサンプル・データベースを開始する設定ファイルを作成します。また、キャッシュを 10 MB に設定し、パーソナル・サーバのこのインスタンスの名前を *Elora* にします。次のように設定ファイルを作成します。

```
# Configuration file for server Elora
-n Elora
-c 10M
samples-dir¥demo.db
```

(先頭に # がある行はコメントとして処理されます。)

samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

ファイルに *sample.txt* という名前を付けます。この設定ファイルを使用してデータベースを開始する場合は、コマンド・ラインで次のように指定します。

```
dbeng10 @sample.txt
```

ここで、単純暗号化を設定に追加します。

```
dbfhide sample.txt encrypted_sample.txt
```

encrypted_sample.txt ファイルを使用してデータベースを開始します。

```
dbsrv10 @encrypted_sample.txt
```


ヒストグラム・ユーティリティ (dbhist)

ヒストグラムを Microsoft Excel チャートに変換します。これには、述部の選択性に関する情報が含まれます。

構文

dbhist [*options*] **-t** *table-name* [*excel-output-filename*]

オプション

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-c *options*	接続パラメータを指定します。「接続パラメータ」 230 ページを参照してください。
-n *colname*	ヒストグラムを関連付けるカラムの名前を指定します。カラムを指定しない場合は、テーブル内のヒストグラムを持っているすべてのカラムが返ります。
-t *table-name*	ヒストグラムを生成するテーブルまたは実体化ビュー (Materialized View) の名前を指定します。
-u *owner*	テーブルまたは実体化ビュー (Materialized View) の所有者を指定します。
excel-output-name	生成される Excel ファイルの名前を指定します。名前を指定しない場合は、Excel から名前の入力を求める [名前を付けて保存] ダイアログが表示されます。

備考

ヒストグラムは ISYSCOLSTAT システム・テーブルに格納され、sa_get_histogram ストアド・プロシージャを使用して取り出せます。ヒストグラム・ユーティリティは、ヒストグラムを Microsoft Excel チャートに変換します。これには、述部の選択性に関する情報が含まれます。ヒストグラム・ユーティリティ (dbhist) は Windows でのみ使用できます。このユーティリティを使用するためには、コンピュータに Excel 97 以降がインストールされている必要があります。

sa_get_histogram ストアド・プロシージャを使用して、ヒストグラムを取得することもできます。「sa_get_histogram システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

文字列カラムに対する述部の選択性を決定するには、ESTIMATE または ESTIMATE_SOURCE 関数を使用してください。文字列カラムからヒストグラムを取り出そうとすると、sa_get_histogram とヒストグラム・ユーティリティがエラーを生成します。

シート名にはカラム名が使用されます。カラム名は、25 文字目以降がトランケートされ、¥、/、?、*、[、]、: (Excel では使用できない文字) はアンダースコア (_) に置き換えられます。チャート名は chart で始まり、上記と同じ命名規則が適用されます。名前が重複した場合は (文字の置換、トランケート、カラム名が chart で始まることなどによって)、重複した名前は使用できないことを示す Excel エラーが発生します。ただし、スプレッドシートは作成され、以前のバージョンで作成された名前 (Sheet1、Chart1 など) が付けられます。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」『SQL Anywhere サーバ - プログラミング』を参照してください。

例

カラムに対してヒストグラムが作成されていると仮定します。次のコマンド (全体を 1 つの行に入力) は、データベース *demo.db* にあるテーブル SalesOrderItems のカラム ProductID の Excel チャートを生成し、それを *histgram.xls* として保存します。

```
dbhist -c "UID=DBA;PWD=sql;DBF=samples-dir¥demo.db" -n ProductID -t SalesOrderItems histgram.xls
```

次の文は、テーブル SalesOrders 内にヒストグラムがあるすべてのカラムを対象とするチャートを生成します。サンプル・データベースがすでに起動されていることを前提とします。この文では、UID=DBA と PWD=sql を使用して接続も行います。出力ファイル名を指定していないので、Excel から入力するように要求されます。

```
dbhist -t SalesOrders -c "UID=DBA;PWD=sql"
```

samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

情報ユーティリティ (dbinfo)

指定したデータベースに関する情報を表示します。

構文

dbinfo [*options*]

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-c *"keyword=value; ⋯"*	接続パラメータを指定します。「接続パラメータ」 230 ページを参照してください。 有効なユーザ ID は情報ユーティリティを実行できますが、ページの使用状況に関する統計を取得するには DBA 権限が必要です。
-o *filename*	指定したファイルに、出力メッセージを書き込みます。
-q	出力メッセージを表示しません。
-u	システム・テーブルやユーザ定義のテーブルを含むすべてのテーブルと実体化ビュー (Materialized View) の使用状況とサイズに関する情報を表示します。 他のユーザがデータベースに接続しておらず、DBA 権限がある場合にのみ、ページ使用状況に関する統計情報を要求できます。ページ使用状況は、sa_table_page_usage システム・プロシージャを使用して取得されます。

備考

dbinfo ユーティリティを使用すると、データベースに関する情報が表示されます。データベースの名前、トランザクション・ログ・ファイルまたはログ・ミラーの名前、ページ・サイズ、照合名とラベル、テーブル暗号化が有効であるかどうかなどの情報がレポートされます。必要に応じて、テーブルの使用状況に関する統計とその詳細を含めることもできます。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

終了コードの詳細については、「ソフトウェア・コンポーネントの終了コード」 『SQL Anywhere サーバ - プログラミング』を参照してください。

初期化ユーティリティ (dbinit)

新しいデータベースを作成します。

構文

dbinit [*options*] *new-database-file*

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-a	CHAR データ型または NCHAR データ型で UCA (ユニコード照合アルゴリズム) を使用している場合 (「-z」と「-zn」を参照)、-a オプションを指定すると、文字列比較では文字によるアクセントの違いが考慮されます (たとえば、e is は é より小さいと見なされます)。デフォルトでは、アクセントは無視されます (この場合、e と é は等しいと見なされます)。すべての基本文字 (アクセントと大文字／小文字の区別を取り除いた文字) が等しい場合は、アクセントが*左から右*へ比較されます。「ユニコード照合アルゴリズム (UCA)」 350 ページを参照してください。
-af	CHAR データ型または NCHAR データ型で UCA を使用している場合 (下記の「-z」と「-zn」を参照)、-af オプションを指定すると、文字列比較では文字によるアクセントの違いが考慮されます (たとえば、e is は é より小さいと見なされます)。デフォルトでは、アクセントは無視されます (この場合、e と é は等しいと見なされます)。すべての基本文字 (アクセントを取り除いた文字) が等しい場合は、フランス語の規則に従ってアクセントが*左から右*へ比較されます。 詳細については、「ユニコード照合アルゴリズム (UCA)」 350 ページを参照してください。

オプション	説明
-b	SQL Anywhere は、文字列について、可変長であり VARCHAR ドメインを使用して格納されている文字列と同じ扱いで、すべての文字列を比較します。これには、固定長の CHAR カラムまたは NCHAR カラムの文字列比較も含まれます。また、値がデータベースに格納されている場合、SQL Anywhere は後続ブランクのトリムや埋め込みは行いません。 デフォルトでは、SQL Anywhere はブランクを意味のある文字として扱います。したがって、値 'a ' (文字 'a' と、後続の 1 つのブランク) は、単一文字列 'a' と等しくありません。不等号比較の照合でも、ブランクは、他の文字と同じように扱われます。 ブランク埋め込みが有効である場合 (dbinit -b オプション)、文字列比較のセマンティックは ANSI/ISO SQL 標準と一層密接になります。ブランク埋め込みが有効であると、SQL Anywhere はどのような比較であっても後続ブランクを無視します。 上に挙げた例では、ブランクを埋め込まれたデータベースで 'a ' を 'a' に対して等号比較すると、TRUE が返されます。ブランクを埋め込まれたデータベースでは、固定長文字列の値は、アプリケーションによってフェッチされたときにブランクで埋め込まれます。このような文字の割り当てが行われたときにアプリケーションが文字列のトランケーション警告を受け取るかどうかは、ansi_blanks 接続オプションによって制御されます。「ansi_blanks オプション [互換性]」 423 ページを参照してください。
-c	このオプションを使用して作成したデータベースでは、すべての値は、比較や文字列操作をするときに大文字と小文字が区別されます。大文字と小文字を区別するデータベースであっても、データベースの識別子については大文字と小文字は区別されません。 このオプションは、ISO/ANSI SQL 標準との互換性を保つために用意されています。デフォルトでは、すべての比較において大文字と小文字が区別されません。

オプション	説明
-dba [*DBA-user*][*,pwd*]	データベースの DBA ユーザに新しい名前を指定すると、そのデータベースにユーザ DBA として接続することはできなくなります。DBA データベース・ユーザに別のパスワードを指定することもできます。パスワードを指定しない場合は、デフォルト・パスワードの sql が使用されます。このオプションを指定しない場合は、パスワードが sql のデフォルト・ユーザ ID DBA が作成されます。 次のコマンドは、いずれも DBA ユーザ名を testuser、デフォルト・パスワードを sql としてデータベースを作成します。 dbinit -dba testuser mydb.db dbinit -dba testuser, mydb.db 次のコマンドは、パスワードが mypwd であるデフォルト・ユーザ ID DBA を使用します。 dbinit -dba ,mypwd mydb.db 次のコマンドは、DBA ユーザをパスワードが mypwd である user1 に変更します。 dbinit -dba user1,mypwd mydb.db パスワードには 7 ビット ASCII 文字を使用することをおすすめします。それ以外の文字を使用すると、サーバがクライアントの文字セットを UTF-8 に変換できない場合、パスワードが機能しないことがあります。
-dbs *size*[**k** | **m** | **g** | **p**]	データベースが使用する領域の事前割り付けを行うと、データベースがあるドライブの空き領域が不足する危険性を小さくすることができます。また、データベース・サイズを拡大する操作は時間を要するため、それが必要となる前にデータベースに保存できるデータの量を増やすことがパフォーマンスの向上につながります。 デフォルトでは、*size* の値はバイト単位となります。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ **k**、**m**、**g** を使用します。ページ数で指定する場合は、単位 **p** を使用します。
-e	データベースの単純暗号化を指定します。このオプションは推奨されなくなりました。代わりに -ea simple を使用してください。

オプション	説明
-ea *algorithm*	このオプションを使用すると、データベース暗号化またはテーブル暗号化 (-et) の設定を指定できます。強力な暗号化を実現するには、FIPS 認定アルゴリズム用に AES または AES_FIPS を設定し、-ek オプションを指定します。AES_FIPS では、AES とは別のライブラリが使用されます。「強力な暗号化」 936 ページを参照してください。 Windows CE では、ARM プロセッサ用に FIPS アルゴリズムがサポートされています。 単純暗号化の場合は、(-ek や -ep を指定するのではなく) -ea simple を指定します。単純暗号化はデータベースの難読化に相当し、データベース・ファイルへの偶然のアクセスが発生した場合にデータが表示されないようにすることだけを目的としています。これにより第三者は、ディスク・ユーティリティを使用してファイルを表示し、データベースのデータを解読することが困難になります。セキュリティを強化するには、強力な暗号化を指定します。 暗号化されていないデータベースを作成するには、-ea オプションを指定しないか (-e、-et、-ep、-et オプションも指定しません)、-ea none を指定します。 -ea オプションを指定しない場合、デフォルトの動作は次のようになります。 ◆ -ea none、-ek (-ep、または -et が指定されない場合) ◆ -ea AES (-ek、または -ep が指定される場合) (-et は指定または指定しない) ◆ -ea simple (-ek や -ep を使用せずに -et を指定する場合) アルゴリズム名の大文字と小文字は区別されません。 次のコマンドは、強力に暗号化されたデータベースを作成し、暗号化キーとアルゴリズムを指定します。 `dbinit -ek "0kZ2o56AK#" -ea AES_FIPS "myencrypteddb.db"` ただし、ファイル圧縮ユーティリティを使用する場合、暗号化を実行したデータベースは、暗号化されていないデータベースほどには圧縮できません。 **別途ライセンスが必要な必須コンポーネント** ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。 「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

オプション	説明
-ek *key*	このオプションを使用すると、コマンドに暗号化キーを直接指定することで、強力に暗号化されたデータベースを作成できます。データベースの暗号化に使用されるアルゴリズムは、-ea オプションで指定した AES または AES_FIPS です。-ek オプションを指定して、-ea オプションを指定しないと、AES アルゴリズムが使用されます。 このオプションを -et とともに指定すると、データベースは暗号化されません。この場合、テーブル暗号化が有効になります。「テーブル暗号化」 943 ページを参照してください。 暗号化キーは保護してください。キーのコピーは、安全な場所に保管してください。キーを紛失すると、データベースにまったくアクセスできなくなり、リカバリも不可能になります。
-ep	このオプションを使用すると、ダイアログ・ボックスに暗号化キーを入力することで、強力に暗号化されたデータベースを作成するように指定できます。クリア・テキストでは暗号化キーを見ることができないようにすることで、高いセキュリティが得られます。 暗号化キーは、正確に入力されたことを確認するために 2 回入力してください。キーが一致しない場合は、初期化は失敗します。 このオプションを -et とともに指定すると、データベースは暗号化されません。この場合、テーブル暗号化が有効になります。 詳細については、「強力な暗号化」 936 ページを参照してください。
-et	このオプションを使用すると、データベースのテーブル暗号化が有効となり、データベース全体を暗号化するのではなく、暗号化されたテーブルを作成することができます。-et オプションを -ek または -ep とともに指定すると、AES アルゴリズムが使用されます。-et だけを指定した場合は、単純暗号化が使用されます。 テーブル暗号化を有効にしただけで、テーブルが暗号化されるわけではありません。データベースを作成した後で、テーブルを個別に暗号化する必要があります。「テーブルの暗号化」 945 ページを参照してください。 テーブルの暗号化が有効な場合、暗号化されたテーブルのテーブル・ページ、関連するインデックス・ページ、テンポラリ・ファイルのページが暗号化されます。さらに、暗号化されたテーブルのトランザクションを含むトランザクション・ログ・ページも暗号化されます。 次の例は、キー abc と暗号化アルゴリズム AES_FIPS を使用する強力なテーブル暗号化を有効にしてデータベース *new.db* を作成します。 `dbinit -et -ek abc -ea AES_FIPS new.db`

オプション	説明
-i	Sybase jConnect JDBC ドライバを使用してシステム・カタログ情報にアクセスするには、jConnect カタログ・サポートをインストールする必要があります (デフォルトでインストールされます)。このオプションは、jConnect システム・オブジェクトを除外したいときに使います。その場合でも、システム情報にアクセスしないかぎり、JDBC を使用できます。必要であれば、Sybase Central または ALTER DATABASE 文を使用して、Sybase jConnect サポートを後から追加することもできます。 詳細については、「jConnect システム・オブジェクトのデータベースへのインストール」『SQL Anywhere サーバ - プログラミング』を参照してください。
-k	デフォルトでは、データベース作成機能は、Watcom SQL (このソフトウェアのバージョン 4 以前) で使用可能なシステム・テーブルとの互換性を保つために、ビュー SYS.SYSCOLUMNS と SYS.SYSINDEXES を生成します。これらのビューは、Sybase Adaptive Server Enterprise の互換性ビュー dbo.syscolumns と dbo.sysindexes と矛盾します。
-l	このオプションを指定すると、dbinit は、推奨する照合順をリストした後停止します。データベースは作成されません。使用可能な照合順のリストは、Sybase Central の [データベース作成] ウィザードに自動的に表示されます。
-le	このオプションを指定すると、dbinit は、使用可能な文字セット・エンコードをリストした後停止します。データベースは作成されません。文字セット・エンコードは、1 つ以上のラベルによって識別されます。エンコードの識別に使用できる文字列があります。表示される各テキスト行には、エンコード・ラベルと、エンコードの識別に使用できる代替ラベルがリストされます。これらのラベルは、SA (SQL Anywhere ラベル)、IANA (Internet Assigned Numbers Authority)、MIME (Multipurpose Internet Mail Extensions)、ICU (International Components for Unicode)、JAVA、または ASE (Adaptive Server Enterprise) の共通カテゴリに分類されます。 代替ラベルを含めて、文字セット・エンコードのリストを参照するには、-le+ オプションを指定します。 初期化ユーティリティが文字セット・エンコードをレポートするとき、必ず、ラベルの SQL Anywhere バージョンもレポートします。たとえば、次のコマンドを使用すると、CHAR データ型の文字セット・エンコード 1250 のレポートが行われます。 `dbinit -ze cp1250 -z uca test.db`
-m *file-name*	トランザクション・ログ・ミラーはトランザクション・ログと同一のコピーで、通常は別のデバイスで管理され、データを確実に保護しています。デフォルトでは、SQL Anywhere はミラー化されたトランザクション・ログを使用しません。

オプション	説明
-n	トランザクション・ログを使わずにデータベースを作成すると、ディスク領域を節約できます。ただし、トランザクション・ログはデータ・レプリケーションに必要です。メディア障害またはシステム障害が発生した場合に備えて、データベース情報のセキュリティを強化します。トランザクション・ログを使用しないデータベースは、通常トランザクション・ログを使用するデータベースより実行速度が遅くなります。
-o *filename*	指定したファイルに、出力メッセージを書き込みます。
-p *page-size*	データベースのページ・サイズには、2048、4096、8192、16384、32768 バイトのいずれかを指定できます。デフォルトは 4096 バイトです。 大規模なデータベースでは、より大きなページ・サイズの方が有利です。たとえば、テーブルのスキャンでは一度にページ全体が読み込まれるため、通常、必要な I/O 操作の回数は少なくなります。ただし、大きなページ・サイズには、より多くのメモリが必要です。ページ・サイズを選択するときは、パフォーマンス・テスト (およびテスト全般) を実行することを強くおすすめします。そして、満足できる結果を得られた最小のページ・サイズを選択します。ほとんどのアプリケーションでは、16 KB または 32 KB のページ・サイズはおすすめしません。常に十分なデータベース・サーバ・キャッシュの確保が可能であり、メモリとディスク領域のパフォーマンス特性に対するトレードオフが調査済である場合以外は、運用システムで 16 KB または 32 KB のページ・サイズは使用しないでください。同じサーバで多数のデータベースが起動される場合は、正しい (かつ適切な) ページ・サイズの選択が特に重要です。 詳細については、次の項を参照してください。 ◆ 「適切なページ・サイズの使用」『SQL Anywhere サーバ - SQL の使用法』 ◆ 「テーブルとページのサイズ」『SQL Anywhere サーバ - SQL の使用法』
-q	クワイエット・モードで実行します (メッセージを表示しません)。
-s	チェックサムは、データベース・ページがディスク上で変更されたかどうかを判断するために使用します。チェックサムを有効にしてデータベースを作成した場合、チェックサムはページがディスクに書き込まれる直前に計算されます。そのページが次にディスクから読み出されるときに、ページのチェックサムが再計算されて、ページに保存されているチェックサムと比較されます。チェックサムが異なる場合は、ディスク上のページが変更されているか、破損しており、エラーが発生します。-s を指定したかどうかにかかわらず、重要なデータベース・ページにはデータベース・サーバによって必ずチェックサムが追加されます。 Windows CE 上に配備するデータベースを作成する場合は、チェックサムを有効にする必要があります。これによって、データベース・ファイルの破損を速やかに検出できます。

オプション	説明
-t *log-name*	トランザクション・ログは、使用しているアプリケーションに関わらず、すべてのユーザが行った変更をデータベース・サーバがその中に記録するファイルです。トランザクション・ログはバックアップとリカバリ (「トランザクション・ログ」 816 ページを参照)、およびデータ・レプリケーションで重要な役割を果たします。トランザクション・ログのファイル名にパスがない場合、トランザクション・ログはデータベース・ファイルと同じディレクトリに保存されます。-t または -n を指定しないで dbinit を実行すると、データベース・ファイルと同じファイル名のトランザクション・ログが作成されますが、拡張子は *.log* になります。
-z *coll* [*collation-tailoring-string*]	照合順は、文字データ型 (CHAR、VARCHAR、LONG VARCHAR) のソートと比較に使用されます。照合は、使用されるエンコード (文字セット) に文字の比較と順序付けに関する情報をもたらすものです。照合は慎重に選択してください。データベースの作成が完了した後で照合を変更するためには、データベースをアンロードし、再ロードする必要があります。照合が指定されていない場合、SQL Anywhere はオペレーティング・システムの言語と文字セットに基づいて照合を選択します。「照合の選択」 352 ページを参照してください。 オプションで、文字列のソートや比較を詳細に制御することを目的に、照合の適合化オプション (*collation-tailoring-string*) を指定できます。これらのオプションは、キーワード=値 の形式で、カッコで囲んで指定して、その後ろに照合名を記述します。次に例を示します。 dbinit -c -z uca(locale=es;case=LowerFirst) spanish2.db これらのキーワードの組み合わせを指定する構文は、CREATE DATABASE 文の COLLATION 句を定義する構文と同じです。「照合の適合化オプション」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。 *collation-tailoring-string* に指定する大文字小文字とアクセント記号の設定は、dbinit (-c、-a、-af) の大文字小文字とアクセント記号の設定よりも優先されます (両方を指定した場合)。 **注意** 照合の適合化オプションを使用して初期化したデータベースは、10.0.1 より前のデータベース・サーバでは起動できません。
-ze *encoding*	-z で指定するほとんどの照合には、エンコード (文字セット) と順序付けの両方が定義されています。そのような照合については、-ze を指定する必要はありません。 -z で指定した照合が UCA (ユニコード照合アルゴリズム) である場合、-ze では CHAR データ型のエンコードとして UTF-8 またはシングルバイト・エンコードを指定できます。デフォルトのエンコードは UTF-8 です。-ze を使用すると、ロケール固有のエンコードを指定し、比較と順序付けに UCA を利用することができます。

オプション	説明
-zn *coll* [*collation-tailoring-string*]	各国の文字データ型 (NCHAR、NVARCHAR、LONG NVARCHAR) のソートと比較に使用する照合順は、-zn オプションで指定します。照合は、使用される UTF-8 エンコード (文字セット) に文字の順序付けに関する情報をもたらすものです。このオプションの値は、UCA (デフォルト) と UTF8BIN です。UTF8BIN は、エンコードが 0x7E を超えるすべての文字のバイナリ順を規定します。dbicu10 と dbicudt10 の DLL がインストールされていない場合、デフォルトの NCHAR 照合は UTF8BIN になります。詳細については、「照合の選択」 352 ページを参照してください。 オプションで、文字列のソートや比較を詳細に制御することを目的に、照合の適合化オプション (*collation-tailoring-string*) を指定できます。これらのオプションは、キーワード=値 の形式で、カッコで囲んで指定して、その後ろに照合名を記述します。次に例を示します。 dbinit -c -zn UCA(case=LowerFirst) sens.db これらのキーワードの組み合わせを指定する構文は、CREATE DATABASE 文の COLLATION 句を定義する構文と同じです。「照合の適合化オプション」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。 *collation-tailoring-string* に指定する大文字小文字とアクセント記号の設定は、dbinit (-c、-a、-af) の大文字小文字とアクセント記号の設定よりも優先されます (両方を指定した場合)。 **注意** 照合の適合化オプションを使用して初期化したデータベースは、10.0.1 より前のデータベース・サーバでは起動できません。

備考

初期化するときに多数のデータベース属性が指定されます。これらの属性は、データベース全体のアンロード、再初期化、再構築以外の方法では、後から変更することはできません。データベース属性には次のものがあります。

- 大文字と小文字の区別の有無
- アクセントの区別
- 句読表記の区別
- 比較における後続ブランクの処理
- ページ・サイズ
- 文字セット・エンコードと照合順
- データベースの暗号化
- テーブル暗号化

たとえば、次のようにして 8192 バイトのページを持つデータベース *test.db* を作成できます。

```
dbinit -p 8192 test.db
```

データベースに大文字小文字またはアクセント記号の区別がある場合は、初期化コマンドに照合の適合化オプションを指定するときに、句読表記の区別でレベル 4 は指定できません。

さらに、初期化するときに、トランザクション・ログとトランザクション・ログ・ミラーを使うかどうかを選択することができます。この選択は、トランザクション・ログ・ユーティリティまたは ALTER DATABASE 文を使用して後で変更できます。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

データベースは、次の方法を使用して作成することもできます。

- Sybase Central の [データベース作成] ウィザードを使用する。「データベースの作成 (Sybase Central の場合)」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- Interactive SQL から CREATE DATABASE 文を使用する。「CREATE DATABASE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

注意
アプリケーションを配備するときは、dbinit ユーティリティを使用してデータベースを作成するために、パーソナル・データベース・サーバ (dbeng10) が必要です。パーソナル・データベース・サーバは、その他のデータベース・サーバが実行されていない場合にローカル・コンピュータで Sybase Central からデータベースを作成する場合にも必要です。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」 『SQL Anywhere サーバ - プログラミング』を参照してください。

例

次のコマンドを実行すると、大文字と小文字を区別するデータベースである *spanish.db* が作成されます。このデータベースでは、非 NCHAR データに対して 1262spa 照合が使用されます。NCHAR データには、UCA 照合が指定されます。この場合、ロケールは es であり、最初に小文字がソートされます。

```
dbinit -c -z 1252spa -zn uca(locale=es;case=LowerFirst) spanish.db
```


Interactive SQL ユーティリティ (dbisql)

SQL コマンドを実行し、データベースに対してコマンド ファイルを実行します。

構文

dbisql [*options*] [*dbisql-command* | *command-file*]

オプション	説明
@data	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-c *"keyword=value; ⋯"*	接続パラメータを指定します。Interactive SQL が接続できない場合は、接続パラメータを入力するダイアログが表示されます。「接続パラメータ」 230 ページを参照してください。
-codepage *codepage*	ファイルを読み込みまたは書き込みするときに使用するコード・ページを指定します。デフォルトのコード・ページは、実行しているプラットフォームのデフォルト・コード・ページです。 たとえば、英語版の Windows XP コンピュータでは、ウィンドウ・ベースのプログラムは 1252 (ANSI) コード・ページを使用します。297 (IBM France) コード・ページを使用して作成されたファイルを Interactive SQL で読み込む場合には、次のオプションを指定します。 -codepage 297 Interactive SQL のデフォルトのコード・ページは、default_isql_encoding オプションでも設定できます。さらに、ENCODING 句を指定して、INPUT 文、OUTPUT 文、または READ 文を発行するときにもコード・ページを選択できます。 詳細については、次の項を参照してください。 ◆ 「推奨文字セットと照合」 361 ページ ◆ 「default_isql_encoding オプション [Interactive SQL]」 612 ページ ◆ 「INPUT 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』 ◆ 「OUTPUT 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』 ◆ 「READ 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』

オプション	説明
-d *delimiter*	コマンド・デリミタを指定します。デリミタを囲む引用符は省略可能ですが、コマンド・シェル自体がデリミタを特別な方法で解釈するときは必ず指定します。 このオプションは、command_delimiter オプションの設定を上書きします。「command_delimiter オプション [Interactive SQL]」 611 ページを参照してください。
-d1	Interactive SQL は、ユーザが明示的に実行したすべての文をコマンド・ウィンドウ (STDOUT) にエコーします。これによって、SQL スクリプトのデバッグ、または Interactive SQL が長文の SQL スクリプトを処理しているときに有用なフィードバックが提供されます。(最後の文字は数値の 1 であり、L の小文字ではありません)。このオプションは、コマンド・プロンプト・モードでのみ使用できます。
-datasource *DSN-name*	接続先の ODBC データ・ソースを指定します。
-f *filename*	*filename* というファイルを (実行せずに) 開きます。ファイル名は引用符で囲むことができますが、ファイル名にスペースが含まれている場合は必ず引用符で囲む必要があります。そのファイルが存在しない場合、またはファイルではなく実際にはディレクトリである場合は、Interactive SQL がエラー・メッセージをコンソールに出力して終了します。ファイル名に完全なドライブとパスの仕様が含まれていないときは、現在のディレクトリが基準として想定されます。
-host *hostname*	データベース・サーバを実行するコンピュータの***ホスト名***または IP アドレスを指定します。現在のコンピュータを意味する localhost を使用できます。
-nogui	Interactive SQL をコマンド・プロンプト・モードで実行します。このとき、ウィンドウ・ベースのユーザ・インタフェースは使用しません。これは、バッチ処理に便利です。*dbisql-command* または *command-file* のいずれかを指定する場合、−nogui が使用されます。 このモードのとき、Interactive SQL は、処理の成功または失敗をプログラム終了コードを設定することで示します。Windows オペレーティング・システムでは、プログラム終了コードに対して環境変数 ERRORLEVEL が設定されます。「ソフトウェア・コンポーネントの終了コード」『SQL Anywhere サーバ - プログラミング』を参照してください。
-onerror { **continue** | **exit** }	コマンド・ファイルから文を読み出し中にエラーが起こった場合の事象を制御します。このオプションは、on_error 設定を上書きします。これは、Interactive SQL をバッチ処理で使用するときに便利です。「on_error オプション [Interactive SQL]」 620 ページを参照してください。

オプション	説明
-port *port-number*	データベース・サーバが実行されているポート番号を指定します。SQL Anywhere のデフォルト・ポート番号は 2638 です。
-q	出力メッセージを表示しません。これは、コマンドまたはコマンド・ファイルを使って Interactive SQL を起動したときにのみ便利です。このオプションを指定した場合、エラー・メッセージは表示されますが、次の情報は表示されません。 ◆ 警告およびその他の致命的でないメッセージ ◆ 結果セットの出力
-x	コマンドをスキャンしますが、実行しません。長いコマンド・ファイルの構文エラーをチェックする場合に有用です。 SQL 文と Interactive SQL コマンドの詳細については、「SQL 言語の要素」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
-ul	Interactive SQL では、接続するデータベースのタイプに応じて使用可能なオプションがカスタマイズされます。Interactive SQL では、デフォルトで SQL Anywhere データベースに接続すると見なされます。-ul オプションを指定すると、デフォルトが Ultra Light データベースに変更されます。デフォルトに設定されているデータベースのタイプに関係なく、[接続] ダイアログのドロップダウン・リストからデータベースのタイプを選択することで、SQL Anywhere または Ultra Light のデータベースに接続できます。 Interactive SQL から Ultra Light データベースへの接続については、「Ultra Light 用 Interactive SQL ユーティリティ (dbisql)」『Ultra Light - データベース管理とリファレンス』を参照してください。

備考

Interactive SQL を使用して、データベースのブラウズ、SQL コマンドの実行、およびコマンド・ファイルの実行を行うことができます。また、影響を受けたローの数、各コマンドに必要な時間、クエリの実行計画、エラー・メッセージに関するフィードバックも提供します。

SQL Anywhere データベースと Ultra Light データベースの両方に接続できます。

dbisql ユーティリティは、Windows、Solaris、Linux、Mac OS X でサポートされています。

dbisql-command を指定すると、Interactive SQL がそのコマンドを実行します。コマンド・ファイル名も指定できます。*dbisql-command* または *command-file* 引数が指定されていないと、Interactive SQL は対話型モードになります。このモードでは、コマンドをコマンド・ウィンドウに入力できます。

次の方法で Interactive SQL を開始できます。

- Sybase Central の [Interactive SQL を開く] メニュー項目を使用する。
- [スタート] メニューから、[プログラム] – [SQL Anywhere 10] – [Interactive SQL] の順に選択する。
- dbisql コマンドを使用する。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。0 以外の終了コードが設定されるのは、(SQL 文またはスクリプト・ファイル名を指定したコマンド・ラインによって) Interactive SQL をバッチ・モードで実行した場合だけです。「ソフトウェア・コンポーネントの終了コード」『SQL Anywhere サーバ - プログラミング』を参照してください。

例

次のコマンドをコマンド・プロンプトで入力すると、ユーザ ID DBA とパスワード sql で、現在のデフォルト・サーバに対してコマンド・ファイル *mycom.sql* が実行されます。コマンド・ファイルにエラーがあった場合は、処理は終了します。

```
dbisql -c "UID=DBA;PWD=sql" -onerror exit mycom.sql
```

次のコマンドをコマンド・プロンプトに 1 行で入力すると、現在のデフォルト・データベースにユーザが追加されます。

```
dbisql -c "UID=DBA;PWD=sql" GRANT CONNECT TO joe IDENTIFIED passwd
```


Interactive SQL ユーティリティ (dbisqlc)

データベースに対して SQL コマンドを実行します。

構文

dbisqlc [*options*] [*dbisqlc-command* | *command-file*]

コマンドをスキャンしますが、実行しません。長いコマンド・ファイルの構文エラーをチェックする場合に有用です。

SQL 文と Interactive SQL コマンドの詳細については、「SQL 言語の要素」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

オプション	説明
-c *"keyword=value*; ⋯*"*	接続パラメータを指定します。Interactive SQL が接続できない場合は、接続パラメータを入力するダイアログが表示されます。「接続パラメータ」 230 ページを参照してください。
-d *delimiter*	コマンド・デリミタを指定します。デリミタを囲む引用符は省略可能ですが、コマンド・シェル自体がデリミタを特別な方法で解釈するときは必ず指定します。 指定したコマンド・デリミタは、現在の dbisqlc セッションのすべての接続で使用できます。
-q	出力メッセージを表示しません。これは、コマンドまたはコマンド・ファイルを使って Interactive SQL を起動したときにのみ便利です。このオプションを指定した場合、エラー・メッセージは表示されますが、次の情報は表示されません。 ◆ 警告およびその他の致命的でないメッセージ ◆ 結果セットの出力
-x	コマンドをスキャンしますが、実行しません。長いコマンド・ファイルの構文エラーをチェックする場合に有用です。 SQL 文と Interactive SQL コマンドの詳細については、「SQL 言語の要素」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

備考

注意
可能なかぎり Interactive SQL を使用することをおすすめします (dbisql コマンドを使用するか、[スタート] メニューから、[プログラム] – [SQL Anywhere 10] – [Interactive SQL] の順に選択する)。dbisqlc ユーティリティでは、Interactive SQL がサポートするすべての機能がサポートされるわけではなく、SQL Anywhere 8、9、10 のデータベースとデータベース・サーバで使用できる機能にもサポートされないものがあるからです。

dbisqlc ユーティリティを使用すると、SQL コマンドを入力したり、Interactive SQL コマンド・ファイルを実行したりすることができます。

dbisqlc ユーティリティは、Windows、NetWare、Mac OS X、UNIX でサポートされています。

dbisqlc-command を指定すると、dbisqlc がそのコマンドを実行します。コマンド・ファイル名も指定できます。*dbisqlc-command* または *command-file* 引数を指定しなかった場合、dbisqlc は対話型モードになります。このモードでは、コマンドをコマンド・ウィンドウに入力できます。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。0 以外の終了コードが設定されるのは、(SQL 文またはスクリプト・ファイル名を指定したコマンド・ラインによって) Interactive SQL をバッチ・モードで実行した場合だけです。「ソフトウェア・コンポーネントの終了コード」『SQL Anywhere サーバ - プログラミング』を参照してください。

例

次のコマンドをコマンド・プロンプトで入力すると、ユーザ ID DBA とパスワード sql で、現在のデフォルト・サーバに対してコマンド・ファイル *mycom.sql* が実行されます。コマンド・ファイルにエラーがあった場合は、処理は終了します。

```
dbisqlc -c "UID=DBA;PWD=sql" mycom.sql
```

次のコマンドをコマンド・プロンプトに 1 行で入力すると、現在のデフォルト・データベースにユーザが追加されます。

```
dbisqlc -c "UID=DBA;PWD=sql" GRANT CONNECT TO joe IDENTIFIED passwd
```


言語選択ユーティリティ (dblang)

SQL Anywhere と Sybase Central によって使用される言語を制御するレジストリ設定のレポートと変更を行います。

構文

dblang [*options*] *language-code*

オプション	説明
-q	クワイエット・モードで実行し、メッセージを表示しません。

言語コード	言語
EN	英語
DE	ドイツ語
ES	スペイン語
FR	フランス語
IT	イタリア語
JA	日本語
KO	韓国語
LT	リトアニア語
PL	ポーランド語
PT	ポルトガル語
RU	ロシア語
TW	中国語 (繁体文字)
UK	ウクライナ語
ZH	中国語 (簡体文字)

備考

このユーティリティは、多言語リソース配備キット (IRDK) をインストール中に選択した場合のみインストールされます。

言語コードを指定しないで dblang ユーティリティを実行すると、現在の設定がレポートされます。次の設定があります。

- **SQL Anywhere**　上述の表に記載された 2 文字の言語コードを保持している HKEY_LOCAL_MACHINE から取得したキー。

  この設定は、SQL Anywhere データベース・サーバから情報メッセージとエラー・メッセージを配信するときに使用される言語リソース・ライブラリを制御します。言語リソース・ライブラリは、*dblgXX10.dll* という形式の名前を持つ DLL です (*XX* は 2 文字の言語コードです)。

  この設定を変更するときは、自分のコンピュータ上に適切な言語リソース・ライブラリがあることを確認してください。

- **Sybase Central**　上述の表に記載された 2 文字の言語コードを保持している HKEY_LOCAL_MACHINE から取得したキー。

  この設定は、Sybase Central と Interactive SQL のユーザ・インタフェース要素を表示するために使用されるリソースを制御します。この設定を有効にするには、SQL Anywhere の適切なローカライズ・バージョンを購入してください。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」 『SQL Anywhere サーバ - プログラミング』を参照してください。

このユーティリティは、設定ファイルからオプションを読み込む ***@data*** *パラメータを受け入れません。*

高速ランチャを使用している場合、言語設定に変更を加えても、Sybase Central や Interactive SQL によって検出されません。これらのツールで新しい言語設定を有効にするには、該当する高速ランチャを無効にして ([ツール] – [オプション] を選択し、[一般] タブで [高速ランチャを有効にする] チェックボックスをオフにします)、言語設定を変更し、ツールを再起動して、高速ランチャを再び有効にする必要があります。

参照

- 「SALANG 環境変数」 313 ページ
- 「高速ランチャの使用」 628 ページ

例

インストール中に多言語リソース配備キット (IRDK) を選択した場合、次のコマンドによって現在の設定を含むダイアログ・ボックスが表示されます。

```
dblang
```


次のコマンドは、設定をフランス語に変更し、前の設定と新しい設定が含まれるダイアログを表示します。

```
dblang FR
```


Log Transfer Manager ユーティリティ (dbltm)

データベースのトランザクション・ログを読み込み、コミットされた変更を Replication Server に送信します。

構文

dbltm [*options*]

オプション

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-A	アップデートをフィルタしません。デフォルトでは、メンテナンス・ユーザが行ったすべての変更はレプリケートされません。-A オプションを設定すると、これらの変更がレプリケートされます。データベースがレプリケート・サイトとプライマリ・サイトの両方として動作する非階層型 Replication Server 環境に便利な場合があります。
-C *config_file*	設定ファイル *config_file* を使用して、LTM 設定を決定します。デフォルトの設定ファイルは *dbltm.cfg* です。「LTM 設定ファイル」 684 ページを参照してください。
-I *interface_file*	(大文字の i) 指定の interfaces ファイルを使用します。interfaces ファイルは、Open Server の接続情報を保持する DSEDIT で作成したファイルです。デフォルトの interfaces ファイルは、*SQL.ini* です。このファイルは、*Sybase* ディレクトリの *ini* サブディレクトリにあります。
-M	これは、リカバリ動作を開始するために使用します。LTM は可能なかぎり早い位置からログの読み込みを開始します。設定ファイルにオフライン・ディレクトリを指定する場合、LTM は最も古いオフライン・ログ・ファイルから読み込みます。
-S *LTM_name*	LTM のサーバ名を指定します。デフォルトの LTM 名は DBLTM_LTM です。LTM 名は、DSEDIT に入力した LTM に対する Open Server 名と対応させます。
-dl	LTM ウィンドウまたはコマンド・プロンプトにすべてのメッセージを表示します。指定されている場合はログ・ファイルにも表示します。

オプション	説明
-ek *key*	このオプションを使用すると、強力に暗号化されているデータベースの暗号化キーをコマンドに直接指定できます。強力に暗号化されたデータベースを扱う場合には、データベースやトランザクション・ログ (オフライン・トランザクション・ログなど) を使用するのに、常に暗号化キーを使用する必要があります。強力な暗号化が適用されたデータベースの場合、-ek または -ep のどちらかを指定します。両方同時には指定できません。強力に暗号化されたデータベースでは、キーを指定しないとコマンドが失敗します。
-ep	このオプションを使用すると、暗号化キーの入力を求めるプロンプトを表示するよう指定できます。このオプションでは、暗号化キーを入力するためのダイアログ・ボックスが表示されます。クリア・テキストでは暗号化キーを見ることができないようにすることで、高いセキュリティが得られます。強力な暗号化が適用されたデータベースの場合、-ek または -ep のどちらかを指定します。両方同時には指定できません。強力に暗号化されたデータベースでは、キーを指定しないとコマンドが失敗します。
-o *filename*	デフォルト (*dbltm.log*) 以外のログ・ファイルを使用します。ログ転送操作による出力メッセージは、このファイルに書き込まれます。
-os *size*	出力ファイルの最大サイズをバイト単位で指定します。最小値は 10000 です。ログ・ファイルのサイズがこの制限値を超えそうになると、ログ・ファイルの名前は *yymmddxx.ltm* に変更されます。*yymmddxx.ltm* の *xx* という値は、その日に作成されるファイルごとに 1 ずつ増加します。
-ot *file*	デフォルト (*dbltm.log*) 以外のログ・ファイルを使用し、LTM の起動時にログ・ファイルをトランケートします (既存の内容はすべて削除されます)。ログ転送操作による出力メッセージはこのファイルに送信されるため、後で検討が可能です。
-q	LTM の起動時にウィンドウを最小化します。
-s	LTL が生成するすべての LTM コマンドを記録します。これは、問題を診断するときだけに使用してください。運用環境ではおすすめしません。これを使用すると、パフォーマンスが大幅に低下します。
-ud	UNIX オペレーティング・システムでは、LTM をデーモンとして実行できます。デーモンとして実行すると、出力はログ・ファイルに記録されます。
-ux	-ux が指定されている場合、dbltm は使用可能な表示を見つけます。たとえば、DISPLAY 環境変数が設定されていなかったり、X-Window Server が実行されていなかったりしたために、使用可能な表示が見つからなかった場合、dbltm は起動できません。Windows では、コンソールは自動的に表示されます。

オプション	説明
-v	デバッグ用に、LTL メッセージ以外のメッセージを表示します。

備考

Log Transfer Manager (LTM) は、「**Replication Agent**」とも呼ばれています。LTM は、プライマリ・サイトとして Replication Server のインストールに関係するすべての SQL Anywhere データベースに必要です。

SQL Anywhere LTM は、データベース・トランザクション・ログを読み込み、コミットした変更を Replication Server に送信します。LTM はレプリケート・サイトには必要ありません。

LTM は、ログ転送言語 (LTL) と呼ばれる言語で Replication Server にコミットされた変更を送信します。

デフォルトでは、LTM は、ログ・ファイル *DBLTM.LOG* を使用して、ステータスとその他のメッセージを保持します。オプションを使用すると、このファイルの名前を変更して、送信されるメッセージの量とタイプを変更できます。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」『SQL Anywhere サーバ - プログラミング』を参照してください。

LTM 設定ファイル

SQL Anywhere と Adaptive Server Enterprise の LTM 設定ファイルは非常に類似しています。この項では、SQL Anywhere LTM 設定ファイルのエントリと、Adaptive Server Enterprise LTM 設定ファイルとの相違について説明します。

LTM が使用する設定ファイルは、-C オプションを使用して指定されます。

LTM 設定ファイルのパラメータ

次の表では、LTM で認識できる各設定パラメータについて説明します。Adaptive Server Enterprise LTM で使用され、SQL Anywhere LTM では使用されないオプションには、「無視される」(この場合は設定ファイルにあっても影響ありません) または「サポートされていない」(この場合は設定ファイルにあるとエラーを起こします) のどちらかが記載されています。

パラメータ	説明
APC_pw	APC_user ログイン名のパスワード。このエントリは、SQL Anywhere LTM 設定ファイルのみにあります。
APC_user	プライマリ・サイトで非同期プロシージャを実行する時に使用するユーザ ID。このユーザ ID には、プライマリ・サイトでのすべての非同期プロシージャに対する適切なパーミッションが必要です。このエントリは、SQL Anywhere LTM 設定ファイルのみにあります。

パラメータ	説明
backup_only	デフォルトは off です。on に設定すると、LTM はバックアップされたトランザクションだけをレプリケートします。
batch_ltl_cmds	on (デフォルト) に設定すると、バッチ・モードの使用が可能です。バッチ・モードは全体的なスループットを向上させますが、応答時間が長くなることがあります。
batch_ltl_sz	batch_ltl_cmds が on のときに、Replication Server に送信される前のバッファに保存されているコマンドの数。デフォルトは 200 です。
batch_ltl_mem	batch_ltl_cmds が on のときに、バッファの内容が Replication Server に送信される前にバッファが使えるメモリの量。デフォルトは 256 KB です。
Continuous	デフォルトは on です。off に設定すると、LTM は、コミットされたデータがレプリケートされると同時に自動的に停止します。
LTM_admin_pw	LTM_admin_user ログイン名のパスワード。
LTM_admin_user	LTM にログインするのに使用するシステム管理者 LTM ログイン名。このパラメータは、LTM を停止するためにログオンしているユーザが正しいログイン名であることを LTM で確認するために必要です。
LTM_charset	LTM が使用する Open Client/Open Server 文字セット。
LTM_language	LTM が使用する Open Client/Open Server 言語。
LTM_sortorder	ユーザ名を比較するために LTM を使用する場合の、Open Client/Open Server ソート順。LTM 文字セット互換の Adaptive Server Enterprise にサポートされているどのソート順でも指定可能です。レプリケーション・システムのすべてのソート順を同一にしてください。 デフォルトのソート順はバイナリ・ソートです。
maint_cmds_to_skip	無視されます。
qualify_table_owners	LTM に対して on に設定すると、テーブル所有者の他にテーブル名とカラム名が Replication Server になっている LTL を送信します。この設定は、レプリケート中のすべてのテーブルに適用されます。レプリケーション作成定義文と一致していることが必要です。デフォルトは off です。
rep_func	on に設定すると、非同期プロシージャ・コール (APC) の使用が可能です。デフォルトは off です。
Retry	失敗した SQL Anywhere データベース・サーバまたは Replication Server への接続を再び行うまでの待ち秒数。デフォルトは 10 秒です。

パラメータ	説明
RS	LTM がログを転送する Replication Server の名前。
RS_pw	RS_user ログイン名に対するパスワード。
RS_source_db	LTM が Replication Server に転送するログのデータベース名。この名前は、Replication Server 接続定義内で定義されているデータベース名と一致させます。ほとんどの場合、RS_Source_db と SQL_database 設定オプションの両方で同じ設定を使用します。
RS_source_ds	LTM が Replication Server に転送するログのサーバ名。この名前は、Replication Server 接続定義内で定義されているサーバ名と一致させます。ほとんどの場合、RS_Source_ds と SQL_server 設定オプションの両方で同じ設定を使用します。
RS_user	Replication Server にログインするために使用する LTM のログイン名。ログイン名には、Replication Server 内の connect source パーミッションの付与が必要です。
scan_retry	トランザクション・ログのスキャン間の LTM の待ち秒数。このパラメータは、Adaptive Server Enterprise LTM のそれとは多少意味が異なります。SQL Anywhere サーバは、レコードがログに届いても、ウェイク・アップせずログもスキャンしません。このため、Adaptive Server Enterprise LTM の scan_retry 値よりも小さい数にできます。
skip_ltl_cmd_err	このパラメータは、LTL コマンド・エラーが発生したときに、Replication Agent に対して処理続行またはシャットダウンを通知します。skip_ltl_cmd_err=on が指定されていると、Replication Agent はエラーの原因となった LTL コマンドを表示し、LTL コマンドをスキップしてレプリケーションを続行します。パラメータが off に設定されている場合は、Replication Agent はエラーの原因となった LTL コマンドを表示し、シャットダウンします。デフォルトでは、このオプションは off に設定されています。
SQL_database	LTM が接続するサーバ SQL_server のプライマリ・サイト・データベース名。リカバリ中の Adaptive Server Enterprise では、LTM が Replication Server に転送するログを持つテンポラリ・データベースです。SQL Anywhere LTM は SQL_log_files パラメータを使用して、オフライン・トランザクション・ログを見つけます。
SQL_log_files	オフライン・トランザクション・ログを保持するディレクトリ。LTM を起動するときにこのディレクトリが必要です。このエントリは、SQL Anywhere LTM 設定ファイルのみにあります。
SQL_pw	SQL_user ユーザ ID に対するパスワード。

パラメータ	説明
SQL_server	LTM が接続するプライマリ・サイト SQL Anywhere サーバの名前。リカバリ中の Adaptive Server Enterprise では、LTM が Replication Server に転送するログを持つテンポラリ・データベースのあるデータ・サーバです。LTM は SQL_log_files パラメータを使用して、オフライン・トランザクション・ログを見つけます。
SQL_user	LTM が RS_source_ds と RS_source_db で指定したデータベースに接続するときに使用するログイン名。

設定ファイルの例

次に、LTM 設定ファイルの例を示します。

```
# This is a comment line
# Names are case sensitive.
SQL_user=SA
SQL_pw=sysadmin
SQL_server=PRIMESV
SQL_database=primedb
RS_source_ds=PRIMEOS
RS_source_db=primedb
RS=MY_REPSERVER
RS_user=sa
RS_pw=sysadmin
LTM_admin_user=DBA
LTM_admin_pw=sql
LTM_charset=cp850
scan_retry=2
SQL_log_files=e:¥logs¥backup
APC_user=sa
APC_pw=sysadmin
```


ログ変換ユーティリティ (dbtran)

トランザクション・ログを SQL コマンド・ファイルに変換します。

構文

データベース・サーバに対して実行する場合。

dbtran [*options*]

トランザクション・ログに対して実行する場合。

dbtran [*options*] [*transaction-log*] [*SQL-file*]

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-a	トランザクション・ログに、最新の COMMIT 以前にすべてのトランザクションが実行したすべての変更が含まれます。最新のコミット以後の変更はトランザクション・ログの中にはありません。 -a が使用されない場合、出力ファイルにはコミットされたトランザクションのみが含まれます。-a を使用する場合、トランザクション・ログにある任意のコミット済みトランザクションに続いて、ROLLBACK 文が出力されます。
-c *"keyword=value*; ⋯*"*	ユーティリティをデータベース・サーバに対して実行する場合、このパラメータは接続文字列を指定します。「接続パラメータ」 230 ページを参照してください。 dbtran の実行には DBA 権限が必要です。
-d	トランザクションは、古いものから新しいものへ順に出力されます。この機能は、主に、データベースのアクティビティを監査するときに使用されます。このコマンドの出力を、データベースに対して適用しないようにしてください。

オプション	説明
-ek *key*	このオプションを使用すると、強力に暗号化されているデータベースの暗号化キーをコマンドに直接指定できます。データベースが強力に暗号化されている場合、データベースまたはトランザクション・ログを使用するには必ず暗号化キーを指定します。 強力な暗号化が適用されたデータベースの場合、-ek または -ep のどちらかを指定します。両方を同時に指定することはできません。強力に暗号化されたデータベースでは、正しいキーを指定しないとコマンドが失敗します。 データベース・サーバに対して実行している (-c オプションを使用している) 場合は、-ek オプションではなく、接続パラメータを使用してキーを指定していることを確認してください。たとえば、次のコマンドは、サーバ **sample** からデータベース *enc.db* についてのトランザクション・ログ情報を取得し、その出力を *log.sql* に保存します。 `dbtran -n log.sql -c "ENG=sample;DBF=enc.db;UID=DBA;PWD=sql;DBKEY=mykey"`
-ep	このスイッチを使用すると、コマンドに暗号化キーの入力を求めるプロンプトを表示するよう指定できます。このオプションでは、暗号化キーを入力するためのダイアログ・ボックスが表示されます。クリア・テキストでは暗号化キーを見ることができないようにすることで、高いセキュリティが得られます。 強力な暗号化が適用されたデータベースの場合、-ek または -ep のどちらかを指定します。両方を同時に指定することはできません。強力に暗号化されたデータベースでは、正しいキーを指定しないとコマンドが失敗します。 データベース・サーバに対して実行している (-c オプションを使用している) 場合は、-ep オプションではなく、接続パラメータを使用してキーを指定していることを確認してください。たとえば、次のコマンドは、サーバ **sample** からデータベース *enc.db* についてのトランザクション・ログ情報を取得し、その出力を *log.sql* に保存します。 `dbtran -n log.sql -c "ENG=sample;DBF=enc.db;UID=DBA;PWD=sql;DBKEY=mykey"`
-f	最終チェックポイント以降に完了したトランザクションだけを出力します。

オプション	説明
-g	auditing データベース・オプションがオンの場合は、監査情報がトランザクション・ログに追加されます。この情報は、このオプションを使用して、出力ファイルにコメントとして含めることができます。「auditing オプション [データベース]」 428 ページを参照してください。 -g オプションは、-a、-d、-t オプションも暗黙で指定したことになります。
-ir *offset1*,*offset2*	2 つの指定オフセット間のトランザクション・ログの一部分を出力します。
-is *source*,…	次にリストするソースの 1 つ以上の操作 (カンマ区切りのリストで指定) によって修正されたローに対する操作を出力します。 ♦ **All** すべてのロー。これはデフォルト設定です。 ♦ **SQLRemote** SQL Remote を使用して修正されたローだけを含みます。SR という短い形式を使用することもできます。 ♦ **RepServer** Replication Agent (LTM) と Replication Server を使用して修正されたローだけを含みます。RS という短い形式を使用することもできます。 ♦ **Local** レプリケートされないローだけを含みます。
-it *owner.table*,…	カンマで区切られた指定のテーブル・リストの操作を出力します。各テーブルは、*owner.table* として指定します。
-j *date/time*	任意の日付または時刻より前の、最新のチェックポイント以降のトランザクションだけを変換します。ユーザは、日付、時刻、日付と時刻を引用符で囲んで引数を指定できます。時刻を省略すると、その日付の開始時刻が使用されます。日付を省略すると、今日の日付が使用されます。日付と時刻には、フォーマット *"YYYY/MMM/DD HH:NN"* を使用できます。
-m	このオプションを使用して、トランザクション・ログを格納するディレクトリを指定します。このオプションは、-n オプションと一緒に使用してください。
-n *filename*	データベース・サーバに対して dbtran ユーティリティを実行するとき、このオプションを使用すると、SQL 文を保持する出力ファイルを指定できます。
-o *filename*	指定したファイルに、出力メッセージを書き込みます。
-q	クワイエット・モードで実行します (メッセージを表示しません)。

オプション	説明
-r	コミットされなかったトランザクションを削除します。これはデフォルトの動作です。
-rsu *username*,…	デフォルトでは、-is オプションは、デフォルトの Replication Server ユーザ名が dbmaint と sa であるとみなします。-rsu オプションを使ってカンマ区切りでユーザ名を指定することで、この前提を無効にできます。
-s	テーブルにプライマリ・キーまたはユニークなインデックスがなく、重複ローがある場合、このオプションを指定しないと、ログ変換ユーティリティは標準ではない FIRST キーワードを使って UPDATE 文を作成します。このオプションを使用すると、FIRST キーワードが省略され、SQL 標準との互換性が保持されます。
-sr	SQL Remote がリモート・サイトに操作を分配する方法を記述するコメントを生成し、出力ファイルに挿入します。
-t	デフォルトでは、トリガが実行する動作はコマンド・ファイルには含まれません。一致するトリガがデータベースにある場合は、コマンド・ファイルがデータベースに対して実行されると、トリガが自動的に動作を実行します。コマンド・ファイルが実行されるデータベースの中に一致するトリガがない場合は、トリガの動作を出力に入れてください。
-u *userid*,…	このオプションを使用すると、指定するユーザのトランザクション・ログだけが出力されるように制限できます。
-x *userid*,…	このオプションを使用すると、指定するユーザ以外のトランザクション・ログが出力されるように制限できます。
-y	このオプションを指定すると、確認メッセージを表示することなく、既存のコマンド・ファイルが自動的に置き換えられます。-q を指定する場合、-y も指定しないと操作は失敗します。
-z	トリガが生成するトランザクションが、出力ファイルの中に単にコメントとして入ります。
transaction-log	変換するログ・ファイル。-c または -m オプションとは一緒に使用できません。
SQL-file	変換した情報を含む出力ファイル。*transaction-log* 専用です。

備考

dbtran ユーティリティは、トランザクション・ログ内の情報を取り出して、それらを一連の SQL 文とコメントとして出力ファイルに入れます。このユーティリティは、次の方法で実行できます。

- **データベース・サーバに対して実行**　このように実行した場合、このユーティリティは標準的なクライアント・アプリケーションです。これは、-c オプションの後に指定された接続文字列を使用してデータベース・サーバに接続し、-n オプションによって指定されたファイルに出力を送ります。この方法での実行には、DBA 権限が必要です。

  次のコマンドは、サーバ **demo10** からログ情報を変換して、出力をファイル *demo.sql* に入れるものです。

  ```
  dbtran -c "ENG=demo10;DBN=demo;UID=DBA;PWD=sql" -n demo.sql
  ```

- **トランザクション・ログ・ファイルに対して実行**　このように実行した場合、ユーティリティは、トランザクション・ログ・ファイルに対して直接作用します。ユーザにこの文を実行する機能を持たせないようにする場合は、トランザクション・ログ・ファイルを一般的なアクセスから保護してください。

  ```
  dbtran demo.log demo.sql
  ```

dbtran ユーティリティが実行されると、トランザクション・ログの初期のログ・オフセットが表示されます。複数のログ・ファイルが作成される場合、順序を決定するにはこの方法が効果的です。

-c を使用する場合、dbtran はオンライン・トランザクション・ログ・ファイルだけでなく、オンライン・トランザクション・ログ・ファイルと同じディレクトリにあるすべてのオフライン・トランザクション・ログ・ファイルを変換しようとします。ディレクトリに複数のデータベース用のトランザクション・ログ・ファイルが含まれている場合、dbtran がエラーを示す場合があります。この問題を回避するには、各ディレクトリに 1 つのデータベースのみのトランザクション・ログ・ファイルが含まれていることを確認します。

トランザクションは複数のトランザクション・ログにまたがる場合があります。トランザクション・ログ・ファイルに複数のログにまたがるトランザクションが含まれている場合、単一トランザクション・ログ・ファイル (たとえば dbtran demo.log) を変換すると、またがっていたトランザクションが失われる場合があります。dbtran によって完全なトランザクションを生成するには、ディレクトリ内のトランザクション・ログ・ファイルで -c または -m オプションを使用します。「複数のトランザクション・ログからのリカバリ」 862 ページを参照してください。

次の方法で、ログ変換ユーティリティにアクセスできます。

- Sybase Central の [ログ・ファイル変換] ウィザードを使用する。
- コマンド・プロンプトで、dbtran コマンドを入力する。バッチまたはコマンド・ファイルへの組み込みには、このユーティリティが便利です。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」 『SQL Anywhere サーバ - プログラミング』を参照してください。

Ping ユーティリティ (dbping)

データベース・サーバを検索し、データベースへの接続をテストします。

構文

dbping [*options*]

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-c *"keyword=value*; ⋯*"*	このオプションでは、dbping の動作を制御する接続パラメータを指定します。接続パラメータを指定しない場合、SQLCONNECT 環境変数が設定されていると、SQLCONNECT 環境変数からの接続パラメータを使用します。「接続パラメータ」 230 ページを参照してください。 次のコマンドを使用して dbping を実行すると、demo10 というデータベース・サーバがすでに実行中である場合、dbping はデータベース demo に接続しようとします。このデータベースがデータベース・サーバ demo10 で実行されていない場合、データベース・サーバは *demo.db* をロードしようとします。demo10 というサーバが見つからない場合、dbping はサーバを自動起動しようとします。 `dbping -d -c "UID=DBA;PWD=sql;ENG=demo10;DBN=demo;DBF=samples-dir¥demo.db"` *samples-dir* の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

オプション	説明

-d	サーバだけではなく、データベースに対しても ping を実行します。
	
	-d オプションを指定すると、dbping はサーバとデータベースに接続できた場合のみ、処理の成功をレポートします。-d オプションを指定しない場合、dbping は、-c オプションによって指定されたサーバを検出すると処理の成功をレポートします。
	
	たとえば、データベース sample を実行する blair という名前のデータベース・サーバがある場合、次のコマンドは成功します。
	
	`dbping -c "ENG=blair;DBN=demo"`
	
	次のコマンドは失敗し、「**データベースへの ping が失敗しました -- 指定されたデータベースが見つかりません。**」というメッセージが表示されます。
	
	`dbping -c "ENG=blair;DBN=demo" -d`

-en	デフォルトでは、 -pc、-pd、-ps のいずれかで指定されたプロパティの値が不明である場合、dbping は NULL を出力し、成功を示すリターン・コードを返して終了します。指定したプロパティが NULL を返した場合に dbping が失敗を示すリターン・コードを返して終了するように変更するには、-en オプションを指定します。このオプションは、-pc、-pd、-ps を指定する場合のみ使用できます。
-l *library*	使用するライブラリを指定します (ファイル拡張子は付けません)。このオプションを使用すると、ODBC ドライバ・マネージャの使用が回避されるので、UNIX オペレーティング・システムでは特に便利です。 たとえば、次のコマンドは、ODBC ドライバを直接ロードします。 `dbping -m -c "DSN=SQL Anywhere 10 Demo" -l dbodbc10` UNIX でスレッド接続ライブラリを使用する場合は、ping ユーティリティのスレッド・バージョンである dbping_r を使用します。
-m	ODBC を使用して接続を確立します。デフォルトでは、このユーティリティは、Embedded SQL インタフェースを使用して接続します。
-o *filename*	指定したファイルに、出力メッセージを書き込みます。

オプション	説明
-pc *property*,…	接続時に、指定した接続のプロパティを表示します。プロパティは、カンマで区切って指定します。このオプションを使用するには、データベース接続を確立するために必要な接続情報を指定してください。「接続レベルのプロパティ」 518 ページを参照してください。 たとえば、次のコマンドは、接続プロパティとして使用できる divide_by_zero_error オプションの設定を表示します。 dbping -c ... -pc divide_by_zero_error
-pd *property*[**@***db-name*],…	接続時に、指定したデータベースのプロパティを表示します。プロパティは、カンマで区切って指定します。「データベース・レベルのプロパティ」 550 ページを参照してください。 たとえば、次のコマンドは、データベースで使用されているページ・サイズを表示します。 dbping -c ... -pd PageSize 必要に応じて、値を取得するデータベースの名前を指定できます。プロパティに **@***db-name* でデータベース名が指定されていない場合は、前のプロパティに指定されたデータベース名が使用されます。 次のコマンドは、データベース mydb で使用されているページ・サイズと照合を表示します。 dbping -c ... -pd PageSize@mydb,Collation
-ps *property*,…	接続時に、指定したデータベース・サーバのプロパティを表示します。プロパティは、カンマで区切って指定します。このオプションを使用するには、データベース接続を確立するために必要な接続情報を指定してください。「サーバ・レベルのプロパティ」 540 ページを参照してください。 たとえば、次のコマンドは、データベース・サーバに対するラインセス・シート数またはプロセッサ数を表示します。 dbping -c ... -ps LicenseCount
-q	クワイエット・モードで実行します (メッセージを表示しません)。

オプション	説明
-s	このオプションでは、dbping を実行しているコンピュータとデータベース・サーバを実行しているコンピュータ間のネットワークのパフォーマンス情報を取得できます。接続速度、遅延時間、スループットの概算値が表示されます。通常は、-c オプションを使用して、目的のサーバ上のデータベースに接続するための接続パラメータを指定します。dbping -s は Embedded SQL 接続でのみ使用できます。-m または -l も指定した場合、-s オプションは無視されます。デフォルトでは、dbping -s は測定する統計ごとに最低 1 秒間、要求のループ処理を実行します。リソースが浪費されることを防ぐために、処理に要する時間にかかわらず、接続と切断は最高 200 回までしか実行されません。低速のネットワークでは、各統計について最低回数の反復処理を実行するために数秒かかることがあります。パフォーマンス統計は概算値であり、クライアント・コンピュータとサーバ・コンピュータの両方がアイドル状態である方が統計値の精度は高くなります。「Embedded SQL 接続のパフォーマンスのテスト」 96 ページを参照してください。
-st *time*	このオプションは -s と同じ働きをします。ただし、-st では、dbping が各統計について要求のループ処理を実行する時間を秒単位で指定できます。このオプションでは、-s を使用した場合より精度の高いタイミング情報を取得できます。「Embedded SQL 接続のパフォーマンスのテスト」 96 ページを参照してください。
-z	このオプションは、Embedded SQL 接続が行われるときのみ使用できます。つまり、このオプションは、-m または -l と組み合わせて使用することはできません。このオプションを使用すると、接続するために使用されたネットワーク通信プロトコルと、他の診断メッセージが表示されます。

備考

dbping ユーティリティは、接続の問題をデバッグするのに役立つツールです。これは、完全な接続文字列か部分的な接続文字列を取り、サーバまたはデータベースが見つかったか、あるいは接続が成功したかどうかを示すメッセージを返します。

このユーティリティは、Embedded SQL または ODBC 接続に対して使用できます。jConnect (TDS) 接続に対しては使用できません。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」『SQL Anywhere サーバ - プログラミング』を参照してください。

再構築ユーティリティ (rebuild)

データベース・ファイルを再構築します。

構文

rebuild *old-database new-database* [*DBA-password*]

備考

バッチ・ファイル、コマンド・ファイル、シェル・スクリプトは、dbunload を使用して *old-database* を *new-database* に再構築します。これは簡単なスクリプトですが、再構築処理の文書化に役立ち、カスタマイズのベースとなります。データベース名は、必ず、拡張子を付けずに指定してください。拡張子 *.db* は自動的に付加されます。

dbunload に -ar オプションを指定すると、Rebuild バッチ・ファイルを使用しなくてもアンロードや再ロードを実行できます。

old-database の DBA ユーザ ID に対するパスワードが最初のパスワード sql でない場合は、*DBA-password* を指定します。

パスワードには 7 ビット ASCII 文字を使用することをおすすめします。それ以外の文字を使用すると、サーバがクライアントの文字セットを UTF-8 に変換できない場合、パスワードが機能しないことがあります。

rebuild はデフォルトのコマンド・ライン・オプションを使用して dbunload、dbinit、Interactive SQL コマンドを実行します。別のオプションが必要な場合は、3 つの手順を別々に実行する必要があります。

データベースの再構築は、Sybase Central で [データベースのアンロード] ウィザードを使用して、アンロード処理の一環として実行することも可能です。「[データベース・アンロード] ウィザードの使用」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」 『SQL Anywhere サーバ - プログラミング』を参照してください。

このユーティリティは、設定ファイルからオプションを読み込む *@data* パラメータを*受け入れません*。

参照

- 「アンロード・ユーティリティ (dbunload)」 739 ページ
- 「初期化ユーティリティ (dbinit)」 662 ページ
- 「Interactive SQL ユーティリティ (dbisql)」 672 ページ

サーバ列挙ユーティリティ (dblocate)

TCP/IP ネットワーク上のデータベース・サーバを検索します。

構文

dblocate [*options*] [*server-name*]

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-d	検出したサーバごとに、サーバ名とアドレスをリストし、その後に各サーバで実行されているデータベースのカンマで区切られたリストを表示します。リストが 160 文字を超える場合は、トランケートされ、末尾に 3 つのピリオド (...) が表示されます。 SQL Anywhere 9.0.2 以前のデータベース・サーバで実行されているデータベースや、起動時に -dh データベース・オプションが指定されたデータベースは、リストされません。「-dh データベース・オプション」 222 ページを参照してください。
-dn *database-name*	指定された名前のデータベースを実行しているサーバのサーバ名とアドレスをリストします。リストが 160 文字を超える場合は、トランケートされ、末尾に 3 つのピリオド (...) が表示されます。 SQL Anywhere 9.0.2 以前のデータベース・サーバで実行されているデータベースや、起動時に -dh データベース・オプションが指定されたデータベースは、リストされません。「-dh データベース・オプション」 222 ページを参照してください。
-dv	検出したサーバごとに、サーバ名とアドレスを表示し、さらに各サーバで実行されているデータベースを別の行にリストします。このリストはトランケートされないため、このオプションを使用すると、-d オプションではトランケートされた部分を確認することができます。 SQL Anywhere 9.0.2 以前のデータベース・サーバで実行されているデータベースや、起動時に -dh データベース・オプションが指定されたデータベースは、リストされません。「-dh データベース・オプション」 222 ページを参照してください。

オプション	説明
-n	コンピュータ名ではなく IP アドレスを出力にリストします。コンピュータ名の検索処理は低速であるため、パフォーマンスの向上につながることがあります。
-o *filename*	指定したファイルに、出力メッセージを書き込みます。
-p *port-number*	指定された TCP/IP ポート番号を使用しているサーバについてのみ、サーバ名とアドレスを表示します。指定する TCP/IP ポート番号は 1 ー 65535 の範囲内にあることが必要です。
-q	クワイエット・モードで実行します (メッセージを表示しません)。
-s *name*	指定された名前のサーバについてのみ、サーバ名とアドレスを表示します。このオプションを使用する場合、-ss オプションは使用できません (両方のオプションを使用すると、サーバがまったく検出されない可能性があります)。
-ss *substr*	指定された部分文字列が名前に含まれるサーバについてのみ、サーバ名とアドレスを表示します。このオプションを使用する場合、-s オプションは使用できません (両方のオプションを使用すると、サーバがまったく検出されない可能性があります)。
-v	デフォルトでは、dblocate の出力において、40 バイトを超えるデータベース・サーバ名はトランケートされます。完全なサーバ名を表示する場合は、-v オプションを指定します。 dblocate を含め、バージョン 9.0.2 以前のクライアントは、名前が 40 バイトを超えるバージョン 10.0.0 以降のデータベース・サーバに接続できません。
server-name	指定した IP アドレスまたはホスト名を持つコンピュータで実行されているデータベース・サーバのみをリストします。たとえば、次のコマンドは、コンピュータ jfrancis 上のサーバを検索します。 dblocate jfrancis -n を指定するかどうかにかかわらず、ホスト名や IP アドレスはどのような形式でもかまいません。たとえば、IP アドレスが 1.2.3.4 であるコンピュータ myhost.mycompany.com で実行されているサーバがあるとします。mycompany.com ドメインから、このコンピュータ上で実行されているサーバだけをリストする場合は、dblocate myhost、dblocate myhost.mycompany.com、dblocate 1.2.3.4 のいずれも使用できます。

備考

サーバ列挙ユーティリティ (dblocate) は、直接接続されているネットワークを介して TCP/IP 上で実行されている SQL Anywhere データベース・サーバを検索し、データベース・サーバとそのア

ドレスのリストを出力します。このリストには代替サーバ名も含まれています。「-sn オプション」 225 ページを参照してください。

ネットワークによっては、dblocate が結果を出力するまでに数秒かかることがあります。

注意
Mac OS X で 2638 以外の TCP/IP ポートを使用しているデータベース・サーバは、-p オプションでその TCP/IP ポートを指定しても、dblocate では検出されません。「ServerPort プロトコル・オプション [PORT]」 283 ページを参照してください。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」 『SQL Anywhere サーバ - プログラミング』を参照してください。

データベース・サーバは、サーバ自体を LDAP サーバとして登録でき、企業内のすべてのサーバを追跡することができます。これにより、クライアントと dblocate は、サーバが WAN または LAN にあるかどうかにかかわらず、IP アドレスを指定せずにファイアウォールを介してサーバを検索できます。LDAP は TCP/IP とともに使用し、ネットワーク・サーバ上でのみ使用されます。「LDAP サーバを使用した接続」 126 ページを参照してください。

同じサーバ名が複数回検索された場合、dblocate は -n オプションが指定されていなくても、各ホストの IP アドレスを表示します。同じサーバ名が見つかるのは、複数の IP アドレスを持つコンピュータ (たとえば、コンピュータに複数のネットワーク・カードがある) でサーバを実行している場合、またはネットワーク・サーバをリモート・コンピュータで実行し、同じ名前を持つパーソナル・サーバをローカル・コンピュータで実行している場合です。

サーバ・ライセンス取得ユーティリティ (dblic)

SQL Anywhere データベース・サーバまたは Mobile Link サーバに、ソフトウェア・ライセンスを適用します。

構文

dblic [*options*] *filename* "*user-name*" "*company-name*"

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-l *type*	ソフトウェアのライセンス契約に記述されているライセンス・モデルと一致するライセンス・タイプを入力します。サポートされるライセンス・タイプは次のとおりです。 ◆ **パーシート**　パーシート・ライセンスは、データベース・サーバに対するクライアント接続の数を制限します。データベース・サーバを実行しているコンピュータ上のプロセッサは、任意の数またはすべてのプロセッサを使用できます。 ◆ **プロセッサ**　プロセッサ・ライセンスは、データベース・サーバが使用できる独立した物理プロセッサの数を制限します。データベース・サーバは、各物理プロセッサを、このライセンス・タイプで使用する 1 つの CPU として扱います。デュアル・コアまたはハイパースレッドのプロセッサを複数のプロセッサとして扱うことはしません。プロセッサ・ライセンスを取得している場合、データベース・サーバに対するクライアント接続の数に制限はありません。
-o *filename*	指定したファイルに、出力メッセージを書き込みます。
-q	クワイエット・モードで実行します (メッセージを表示しません)。
-u *license-number*	ライセンスされたユーザまたはプロセッサの総数。ライセンスを追加する場合でも、追加ライセンスの数***ではなく***、総数を指定します。
filename	ライセンスされているパーソナル・データベース・サーバ、ネットワーク・データベース・サーバ、または Mobile Link サーバのサーバ実行プログラムまたはライセンス・ファイルのパスと名前を入力します。 ライセンス・ファイル名だけを入力することで、サーバ実行プログラムの現在のライセンス情報を表示できます。

オプション	説明
user-name	ライセンスのユーザ名。この名前は、起動時にサーバ・メッセージ・ウィンドウに表示されます。名前にスペースが含まれている場合、二重引用符で囲んでください。
company-name	ライセンスの会社名。この名前は、起動時にサーバ・メッセージ・ウィンドウに表示されます。名前にスペースが含まれている場合、二重引用符で囲んでください。

備考

サーバ・ライセンス取得ユーティリティでは、SQL Anywhere データベース・サーバまたは Mobile Link サーバにライセンス・ユーザまたはライセンス・プロセッサを追加できます。このユーティリティは、ライセンス契約に基づいて、許可されたライセンス・ユーザ数またはライセンス・プロセッサ数の範囲内で使用してください。このコマンドの実行によって、ライセンスが付与されることは*ありません*。

このユーティリティでは、パーソナル・データベース・サーバ、ネットワーク・データベース・サーバ、または Mobile Link サーバによって起動時に表示されるユーザ名や会社名を変更することもできます。

また、ライセンス・ファイル名だけを入力することで、パーソナル・データベース・サーバやネットワーク・データベース・サーバを起動しないで、現在のライセンス情報を表示することも可能です。

ライセンス情報は、サーバの実行プログラムと同じディレクトリにある *.lic* ファイルに格納されます。サーバは、実行中の実行プログラムと同じベース・ファイル名を持つ *.lic* ファイルを探します。たとえば、データベース・サーバの実行プログラムが *myserver.exe* という名前の場合、サーバは *myserver.lic* という名前のライセンス・ファイルを探します。デフォルトでは、次の名前が使用されます。

実行プログラム	ライセンス・ファイル名
SQL Anywhere パーソナル・データベース・サーバ (dbeng10)	*dbeng10.lic*
SQL Anywhere ネットワーク・データベース・サーバ (dbsrv10)	*dbsrv10.lic*
Mobile Link サーバ (mlsrv10)	*mlsrv10.lic*

サーバを起動するときに、対応する *.lic* ファイルが使用できない場合は、サーバは起動されません。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」『SQL Anywhere サーバ - プログラミング』を参照してください。

UNIX では、データベース・サーバの実行プログラムはデフォルトで書き込みができないので、サーバ・ライセンス取得 (dblic) ユーティリティを使用すると失敗します。実行プログラムは、サーバ・ライセンス取得ユーティリティを使用する前に (たとえば chmod +w を使用して) 書き込み可能にしてください。

サーバ・ライセンス取得ユーティリティは、NetWare ではサポートされていません。NetWare で SQL Anywhere 実行プログラムをライセンスするには、NetWare 上の SQL Anywhere データベース・サーバが実行中でないことを確認してから、SQL Anywhere ソフトウェアが保存されている NetWare ボリュームにアクセスできる Windows コンピュータからサーバ・ライセンス取得ユーティリティを実行してください。

SQL Anywhere ライセンス取得の詳細については、http://www.ianywhere.com/products/sa_licensing.html を参照してください。

例

次のコマンドをデータベース・サーバの実行プログラムと同じディレクトリで実行すると、ユーザ名 Sys Admin、会社名 My Co という設定で、50 ユーザのライセンスが、Windows ネットワーク・データベース・サーバに適用されます。コマンドは、1 行に入力してください。

```
dblic -l perseat -u 50 dbsrv10.lic "Sys Admin" "My Co"
```

ライセンスの適用が成功すると、画面に次のメッセージが表示されます。

```
ライセンスされるノード: 50
ユーザ: Sys Admin
会社: My Co
```

次のコマンドは、データベース・サーバのライセンスについての情報を返します。

```
dblic dbsrv10.lic
```

Windows 用サービス・ユーティリティ (dbsvc)

SQL Anywhere サービスの作成、変更、削除を行います。

構文

dbsvc [*modifier-options*] **-d** *svc*

dbsvc [*modifier-options*] **-g** *svc*

dbsvc [*modifier-options*] **-l**

dbsvc [*modifier-options*] **-u** *svc*

dbsvc [*modifier-options*] *creation-options* **-w** *svc details*

dbsvc [*modifier-options*] **-x** *svc*

details:

<full-executable-path> [*options*]

主要オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードやその他の情報を保護する場合は、ファイル非表示ユーティリティを使用すると、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-d *service-name*	サービス・リストから指定のサービスを削除します。-y を指定すると、確認メッセージを表示せずにサービスを削除します。
-g *service-name*	パスワード以外のサービスの定義をリストします。
-l	使用できる SQL Anywhere サービスをリストします。
-u *service-name*	*service-name* という名前のサービスを起動します。

主要オプション	説明
-w *executable parameters*	新しいサービスを作成するか、同名のサービスが存在する場合はそれを上書きします。-y を指定すると、確認メッセージを表示せずに既存のサービスを上書きします。 サービスとして使用する実行プログラムのフル・パスを指定します。これは、その下でサービスが実行されるアカウントのパスの中に適切な SQL Anywhere インストール・ディレクトリがない場合があるからです。 作成するサービスに適したパラメータを指定します。 次の項を参照してください。 ◆ **dbsrv10 と dbeng10** 「SQL Anywhere データベース・サーバ」 138 ページ ◆ **mlsrv10** 「Mobile Link サーバのオプション」 『Mobile Link - サーバ管理』 ◆ **dbmlsync** 「dbmlsync 構文」 『Mobile Link - クライアント管理』 ◆ **dblsn** 「Listener 構文」 『Mobile Link - サーバ起動同期』 ◆ **dbremote** 「Message Agent」 『SQL Remote』 ◆ **dbns** 「SQL Anywhere Broadcast Repeater ユーティリティ (dbns10)」 715 ページ ◆ **dbltm** 「Log Transfer Manager ユーティリティ (dbltm)」 682 ページ
-x *service-name*	*service-name* という名前のサービスを停止します。

作成オプション	説明
-a *acct*	すべてのサービスは、Windows アカウントの下で実行されます。作成したアカウントの下で実行する場合は、-a オプションでアカウントを指定し、-p オプションでパスワードを指定します。 サービスとしてログインする権限は、LocalSystem を除くすべてのアカウントに必要です。アカウントのサービスとしてログインする権限が有効でない場合は、有効にするよう要求されます。-y オプションも指定すると、dbsvc はプロンプトを表示することなくサービスとしてログインする権限を付与しようとします。-y オプションを指定しないで -q オプションだけを指定すると、サービスとしてログインする権限を有効にするよう要求するプロンプトは表示されず、dbsvc は失敗します。

作成オプション	説明
-as	すべてのサービスは、Windows アカウントの下で実行されます。-as を指定すると、サービスは Windows LocalSystem アカウントの下で実行されます。パスワードは必要ありません。-a または -as のいずれかを必ず使用してください。
-i	アイコンが表示され、これをダブルクリックするとサーバ・メッセージ・ウィンドウが表示されます。
-p	サービスが実行されるアカウントのパスワードを指定するには、このオプションを -a オプションと一緒に使用します。
-rg *dependency*,…	作成するサービスが起動できるよう事前に起動しておく必要のある 1 つ以上の起動順序グループを指定します。
-rs *dependency*,…	リストに含まれるすべてのサービスが起動してから、作成したサービスの起動を許可します。
-s *startup*	SQL Anywhere サービスの起動時の動作を設定します。Automatic、Manual、または Disabled という起動時の動作を設定できます。デフォルトは Manual です。
-sd *description*	このオプションを使用して、サービスの説明を表示します。説明は、Windows のサービス マネージャに表示されます。
-sn *name*	このオプションを使用して、サービスの名前を指定します。この名前は、Windows のサービス マネージャに表示されます。-sn オプションを指定しない場合、デフォルトのサービス名は SQL Anywhere - *svc* です。たとえば、次のサービスにはデフォルトで SQL Anywhere - myserv という名前が付けられます。 dbsvc -as -w myserv "c:¥Program Files¥SQL Anywhere 10¥win32¥dbeng10.exe" サービス名 myserv が Windows のサービス・マネージャに表示されるようにするには、次のコマンド (全体を 1 行に入力) を実行する必要があります。 dbsvc -as -sn myserv -w myserv "c:¥Program Files¥SQL Anywhere 10¥win32¥dbeng10.exe"

作成オプション	説明
-t *type*	このサービスのタイプを指定します。次のタイプから選択できます。 ◆ **Network** SQL Anywhere ネットワーク・データベース・サーバ (dbsrv10)。「SQL Anywhere データベース・サーバ」 138 ページを参照してください。 ◆ **Standalone** SQL Anywhere パーソナル・データベース・サーバ (dbeng10)。「SQL Anywhere データベース・サーバ」 138 ページを参照してください。 ◆ **DBLTM** SQL Anywhere Log Transfer Manager (LTM)。このサービス・タイプには **LTM** を指定することも可能です。「Log Transfer Manager ユーティリティ (dbltm)」 682 ページを参照してください。 ◆ **DBRemote** SQL Remote Message Agent (dbremote)。「Message Agent」 『SQL Remote』を参照してください。 ◆ **Mobile Link** Mobile Link サーバ (mlsrv10)。「mlsrv10 の構文」 『Mobile Link - サーバ管理』を参照してください。 ◆ **dbmlsync** Mobile Link 同期クライアント (dbmlsync)。「dbmlsync 構文」 『Mobile Link - クライアント管理』を参照してください。 ◆ **DBNS** SQL Anywhere Broadcast Repeater (dbns10)。「SQL Anywhere Broadcast Repeater ユーティリティ (dbns10)」 715 ページを参照してください。 ◆ **dblsn** Listener ユーティリティ (dblsn)「Listener 構文」 『Mobile Link - サーバ起動同期』を参照してください。 すべてのサービス・タイプのデフォルト設定は Standalone です。

変更オプション	説明
-cm	サービスの作成に使用するコマンドを表示します。このオプションを使用すると、作成コマンドをファイルに出力できます。このファイルを使用して、別のコンピュータにサービスを追加したり、変更が加えられたサービスを元の状態にリストアしたりすることができます。-cm とともに -g オプションまたは -l オプションを指定しないとコマンドは失敗します。-g を指定すると、指定のデータ・ソースの作成コマンドが表示されるのに対し、-l を指定するとすべてのデータ・ソースの作成コマンドが表示されます。 指定のサービスが存在しない場合は、サービスを削除するコマンドが生成されます。たとえば、コンピュータに service_1 が存在しない場合、dbsvc -cm -g service_1 は次のコマンドを返して service_1 サービスを削除します。 `dbsvc -y -d "service_1"` サービスが LocalSystem アカウントを使用しない場合は、パスワードを取り出すことはできないため、生成されるコマンドにはパスワードは含まれません。-a *user* -p *password* を使ってサービスを作成した場合、出力には -a *user* のみが含まれます。
-o *log-file*	サービス・ユーティリティ (dbsvc) の出力を指定されたファイルに書き込みます。-o オプションは、必ず-d、-g、-l、-u、-x の前に指定してください。dbsvc とサービスとして実行する実行ファイル (データベース・サーバなど) の両方に指定すると、それぞれについて ログ・ファイルが作成されます。次に例を示します。 `dbsvc -o out1.txt -y -as -w mydsn install-dir¥win32¥dbsrv10 -n mysrv -o c:¥out2.txt` この場合は、dbsvc の出力が *out1.txt* に書き込まれ、データベース・サーバの出力は *c:¥out2.txt* に書き込まれます。
-q	サーバ・メッセージ・ウィンドウにメッセージを表示しません。-q を指定する場合は、-o オプションを使用してメッセージを書き込むファイルを指定するようおすすめします。既存のサービスの変更時または削除時にこのオプションを指定する場合、-y も指定しないと操作は失敗します。
-y	確認メッセージを表示することなく、処理を実行します。このオプションは、-w または -d オプションと一緒に使用できます。既存のサービスの変更時または削除時に -q を指定する場合、-y も指定しないと操作は失敗します。

備考

サービスは、一連のオプションを使ってデータベース・サーバやその他のアプリケーションを実行します。このユーティリティでは、Windows 上で動作している SQL Anywhere サービスを包括的な方法で管理できます。サービス・ユーティリティは、Sybase Central の [サービス作成] ウィザードと同じ機能を提供します。

サービス・ユーティリティを使用するには、ローカル・コンピュータ上で Administrators グループのメンバである必要があります。

サービス・ユーティリティにアクセスするには、次の方法があります。

- Sybase Central の [サービス作成] ウィザードを使用する。「Windows サービスの作成」 39 ページ を参照してください。
- コマンド・プロンプトで、dbsvc コマンドを入力する。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」 『SQL Anywhere サーバ - プログラミング』を参照してください。

参照

- 「Windows サービスの概要」 38 ページ

例

指定のサーバを指定のパラメータで起動する、myserv という名前のパーソナル・サーバ・サービスを作成します。このサーバは LocalSystem ユーザとして実行されます。

```
dbsvc -as -w myserv "c:¥Program Files¥SQL Anywhere 10¥win32¥dbeng10.exe" -n myeng -c 8m "c:¥temp
¥mysample.db"
```

mynetworkserv という名前のネットワーク・サーバ・サービスを作成します。このサーバはローカル・アカウントで実行され、コンピュータの起動時に自動的に起動します。

```
dbsvc -as -s auto -t network -w mynetworkserv "c:¥Program Files¥SQL Anywhere 10¥win32¥dbsrv10.exe"
-x tcpip -c 8m "c:¥temp¥mysample.db"
```

サービス myserv についての詳細をすべてリストします。

```
dbsvc -g myserv
```

myserv という名前のサービスを、確認メッセージを表示せずに削除します。

```
dbsvc -y -d myserv
```

Workstation サービスと TDI グループに依存するサービスを作成します。

```
dbsvc -rs Workstation -rg TDI -w ...
```

mysyncservice という名前のサービスを作成します。

```
dbsvc -as -s manual -t dbmlsync -w mysyncservice "c:¥Program Files¥SQL Anywhere 10¥win32
¥dbmlsync.exe" -c "SQL Anywhere 10 CustDB"
```

service_1 サービスを作成するためのコマンドを生成し、それを *restoreservice.bat* というファイルに出力します。

```
dbsvc -cm -g service_1 > restoreservice.bat
```

restoreservice.bat ファイルには以下が含まれています。

```
dbsvc -t Standalone -s Manual -as -y -w "service_1"
"c:¥Program Files¥SQL Anywhere 10¥win32¥dbeng10.exe"
```

手動で起動される Mobile Link Listener サービスを作成します。


```
dbsvc -as -i -w myListener "c:¥Program Files¥SQL Anywhere 10¥win32¥dblsn.exe" "@c:¥temp¥dblsn.opt"
```

myListener サービスを起動します。

```
dbsvc -u myListener
```

myListener サービスを停止します。

```
dbsvc -x myListener
```

Linux 用サービス・ユーティリティ (dbsvc)

SQL Anywhere サービスの作成、変更、削除を行います。

構文

dbsvc [*modifier-options*] **-d** *svc*

dbsvc [*modifier-options*] **-g** *svc*

dbsvc [*modifier-options*] **-l**

dbsvc [*modifier-options*] **-u** *svc*

dbsvc [*modifier-options*] *creation-options* **-w** *svc details*

dbsvc [*modifier-options*] **-x** *svc*

details:

full-executable-path [*options*]

主要オプション	説明
-d *service-name*	サービス・リストから指定のサービスを削除します。-y を指定すると、確認メッセージを表示せずにサービスを削除します。
-g *service-name*	サービスの定義を表示します。
-l	使用できる SQL Anywhere サービスをリストします。
-u *service-name*	*service-name* という名前のサービスを起動します。
-w *executable parameters*	新しいサービスを作成するか、同名のサービスが存在する場合はそれを上書きします。-y を指定すると、確認メッセージを表示せずに既存のサービスを上書きします。 作成するサービスに適したパラメータを指定します。 次の項を参照してください。 ◆ **dbsrv10 と dbeng10** 「SQL Anywhere データベース・サーバ」 138 ページ ◆ **mlsrv10** 「Mobile Link サーバのオプション」 『Mobile Link - サーバ管理』 ◆ **dbmlsync** 「dbmlsync 構文」 『Mobile Link - クライアント管理』 ◆ **dbremote** 「Message Agent」 『SQL Remote』
-x *service-name*	*service-name* という名前のサービスを停止します。

作成オプション	説明
-a *acct*	すべてのサービスは、Linux アカウントの下で実行されます。独自に作成したアカウントで実行する場合は、-a オプションでアカウントを指定する必要があります。 サービスとしてログインする権限は、デーモン・アカウントを除くすべてのアカウントに必要です。
-as	すべてのサービスは、Linux アカウントの下で実行されます。-as を指定すると、サービスは Linux デーモン・アカウントの下で実行されます。パスワードは必要ありません。-a または -as のいずれかを必ず使用してください。
-t *type*	このサービスのタイプを指定します。次のタイプから選択できます。 ♦ **ネットワーク** SQL Anywhere ネットワーク・データベース・サーバ (dbsrv10)。「SQL Anywhere データベース・サーバ」 138 ページを参照してください。 ♦ **Standalone** SQL Anywhere パーソナル・データベース・サーバ (dbeng10)。「SQL Anywhere データベース・サーバ」 138 ページを参照してください。 ♦ **DBRemote** SQL Remote Message Agent (dbremote)。「Message Agent」『SQL Remote』を参照してください。 ♦ **Mobile Link** Mobile Link サーバ (mlsrv10)。「mlsrv10 の構文」『Mobile Link - サーバ管理』を参照してください。 ♦ **dbmlsync** Mobile Link 同期クライアント (dbmlsync)。「dbmlsync 構文」『Mobile Link - クライアント管理』を参照してください。 すべてのサービス・タイプのデフォルト設定は Standalone です。

変更オプション	説明
-cm	サービスの作成に使用するコマンドを表示します。このオプションを使用すると、作成コマンドをファイルに出力できます。このファイルを使用して、別のコンピュータにサービスを追加したり、変更が加えられたサービスを元の状態にリストアしたりすることができます。-cm とともに -g オプションまたは -l オプションを指定しないとコマンドは失敗します。-g を指定すると、指定のデータ・ソースの作成コマンドが表示されるのに対し、-l を指定するとすべてのデータ・ソースの作成コマンドが表示されます。
-q	コンソールにメッセージを表示しません。既存のサービスの変更時または削除時にこのオプションを指定する場合、-y も指定しないと操作は失敗します。

変更オプション	説明
-y	確認メッセージを表示することなく、処理を実行します。このオプションは、-w または -d オプションと一緒に使用できます。既存のサービスの変更時または削除時に -q を指定する場合、-y も指定しないと操作は失敗します。

Linux 固有のオプション	説明
-od	システム情報ファイルのロケーションを指定します (必要な場合)。
-pr	Linux プロセスのナイスレベルを設定します。
-rl	サービスを起動するランレベルを指定します。
-rs	サービスを作成するときのサービス依存性を指定します。 リストに含まれるすべてのサービスが起動してから、作成したサービスの起動を許可します。Linux サービスはベース・サービスに依存します。たとえば、サービス A がサービス B に依存し、サービス B はサービス C に依存している場合、SQL Anywhere はサービス C を起動しようとしません。
-status	サービスの実行ステータスを返します。

備考

サービスは、一連のオプションを使ってデータベース・サーバやその他のアプリケーションを実行します。このユーティリティを使用して、Linux 上で動作している SQL Anywhere サービスを包括的な方法で管理できます。

通常、サービスの作成環境と実行環境は異なるため、サービスの作成時にはデータベース・ファイルに完全に修飾された名前を付けることをおすすめします。また、データ・ソース名にはスペースを入れないでください。

dbsvc ユーティリティでは、ほとんどの Linux サービスと同じように、サービス・ファイルが */etc/init.d* に作成されます。サービス名は、**SA_***service-name* の形式で指定します。たとえば、SA_myserv という名前のサービスを作成した場合、このサービスを起動するには、次のようなコマンドを発行します。

```
/etc/init.d/SA_myserv start
```

次のコマンドは、サービスのステータスを取得します。

```
/etc/init.d/SA_myserv status
```

次のコマンドは、サービスの使用状況に関する情報を返します。

```
/etc/init.d/SA_myserv
```

例

指定のサーバを指定のパラメータで起動する、myserv という名前のパーソナル・サーバ・サービスを作成します。このサーバは LocalSystem ユーザとして実行されます。

```
dbsvc -as -w myserv -n myeng -c 8m "/tmp/demo.db"
```


mynetworkserv という名前のネットワーク・サーバ・サービスを作成します。このサーバはローカル・アカウントで実行され、コンピュータを再起動すると自動的に起動します。

```
dbsvc -as -t network -w mynetworkserv -x tcpip -c 8m "/tmp/demo.db"
```

サービス myserv についての詳細をすべてリストします。

```
dbsvc -g myserv
```

myserv という名前のサービスを、確認メッセージを表示せずに削除します。

```
dbsvc -y -d myserv
```

mysyncservice という名前のサービスを作成します。

```
dbsvc -as -t dbmlsync -o syncinfo.txt -w mysyncservice -c "/tmp/CustDB.db"
```

service_1 サービスを作成するコマンドを生成し、そのコマンドをコンソールに出力します。

```
dbsvc -cm -g service_1
```

コンソールには次のように表示されます。

```
'dbsvc -t Standalone -as -y -w "service_1" -n'
```

dbsvc を使用してサービスを起動します。

```
dbsvc -u myserv
```

dbsvc を使用してサービスを停止します。

```
dbsvc -x myserv
```

dbsvc を使用してサービスのステータスを取得します。

```
dbsvc -status myserv
```

SQL Anywhere Broadcast Repeater ユーティリティ (dbns10)

SQL Anywhere クライアントは、UDP ブロードキャストは通常到達しないような、別のサブネット上やファイアウォールを介して実行している SQL Anywhere データベース・サーバを検索できます。

構文

dbns10 [*options*] [*address ...*]

オプション

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードやその他の情報を保護する場合は、ファイル非表示ユーティリティを使用すると、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-ap *port*	データベース・サーバで使用されるポート番号を指定します。デフォルトのポート番号は 2638 です。
-m *ip*	DBNS プロセスが実行されているコンピュータの IP アドレスを指定します。このパラメータは、複数の IP アドレスがあるコンピュータでは必須です。このアドレスは IPv4 アドレスであることが必要です。
-o *file*	SQL Anywhere Broadcast Repeater のメッセージ・ウィンドウに表示される出力を指定されたファイルに書き込みます。
-p *port*	DBNS Broadcast Repeater が使用するポート番号を指定します。デフォルトは 3968 です。サブネット間にファイアウォールがある場合は、クライアント／サーバ通信用のポート 2638 に加え、Broadcast Repeater ユーティリティが DBNS プロセス間の TCP 接続に使用するポートも開く必要があります。
-q	クワイエット・モードで実行します (メッセージを表示しません)。
-s	このオプションを指定して SQL Anywhere Broadcast Repeater を起動すると、DBNS プロセスは別の DBNS プロセスが同じサブネット上で実行されていないかどうかをチェックし、既存の DBNS プロセスが見つかった場合は、エラーを返してから停止します。

オプション	説明
-x	指定されたホスト上の DBNS プロセスを停止します。IP アドレスとホスト名のどちらでも指定できます。
-z	デバッグ・モードで実行すると、受信または送信された各 SQL Anywhere ブロードキャスト・パケットに対応する行が SQL Anywhere Broadcast Repeater のメッセージ・ウィンドウに表示されます。デバッグ出力が冗長であるため、接続上の問題がある場合のみデバッグ・モードを使用してください。
address	DBNS プロセスを実行中または将来実行する他のコンピュータの IP アドレスまたはホスト名を指定します。これによって、DBNS プロセスが互いを検出し、既知のデータベース・サーバや他の DBNS プロセスの情報を交換できます。

備考

SQL Anywhere Broadcast Repeater を使用すると、SQL Anywhere クライアントにおいて、HOST 接続パラメータや LDAP を使用しなくても、他のサブネット上で通常は UDP ブロードキャストが到達できないファイアウォールを介して動作している SQL Anywhere データベース・サーバを検出することができます。

address には、IP アドレスとコンピュータ名のどちらでも指定できます。複数のアドレスを指定する場合は、スペースで区切ります。

このユーティリティは、32 ビットおよび 64 ビットのすべての Windows プラットフォームと、サポートされるすべての UNIX プラットフォームで使用できます。

SQL Anywhere Broadcast Repeater を使用するためには、クライアントとデータベース・サーバで SQL Anywhere 9.0.2 以降が実行されている必要があります。

警告
dbns10 ユーティリティを SQL Anywhere データベース・サーバと同じコンピュータ上で実行しないようおすすめします。dbns10 またはデータベース・サーバが UDP ブロードキャストを受信しなくなる可能性があります。

参照

- 「SQL Anywhere Broadcast Repeater を使用したデータベース・サーバの検出」 93 ページ

例

サブネット 10.50.83.255 と 10.50.125.255 のコンピュータがブロードキャストを使用して接続できるようにする場合を想定します。10.50.83.255 サブネット上のコンピュータ (10.50.83.114 のコンピュータ A) と 10.50.125.255 サブネット上のコンピュータ (10.50.125.103 のコンピュータ B) が必要です。

両方のコンピュータで dbns10 を実行し、それぞれ他方のコンピュータの IP アドレスを渡します。コンピュータ A では次のコマンドを実行します。

```
dbns10 10.50.125.103
```

コンピュータ B では次のコマンドを実行します。

```
dbns10 10.50.83.114
```

いずれかのコンピュータに複数の IP アドレスがある場合は、-m オプションを使用してローカル IP アドレスも指定する必要があります。たとえば、コンピュータ A では次のようなコマンドを使用します。

```
dbns10 -m 10.50.83.114  10.50.125.103
```


SQL Anywhere コンソール・ユーティリティ (dbconsole)

データベース・サーバ接続の管理機能とモニタリング機能を提供します。

構文

dbconsole [*options*]

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-c *"keyword=value; ⋯"*	接続パラメータを指定します。「接続パラメータ」 230 ページを参照してください。
-datasource *DSN-name*	接続先の ODBC データ・ソースを指定します。このオプションを使用するために、iAnywhere JDBC ドライバを使用する必要はありません。
-host *hostname*	データベース・サーバを実行するコンピュータの***ホスト名***または IP アドレスを指定します。現在のコンピュータを意味する localhost を使用できます。
-port *port-number*	データベース・サーバが実行されているポート番号を指定します。SQL Anywhere のデフォルト・ポート番号は **2638** です。

備考

SQL Anywhere コンソールでは、クライアント・コンピュータからサーバをモニタできます。このユーティリティはネットワーク・サーバ・モニタとも呼ばれます。これを使うと、使用しているネットワーク上でデータベース・サーバにログオンしているユーザを追跡できます。また、ローカル・クライアント画面へのサーバとクライアント統計の表示、ユーザの切断、データベース・サーバの設定を行うことができます。SQL Anywhere コンソールは、複数の接続に関する情報を表示できます。

♦ データベースからユーザを切断するには、次の手順に従います。

1. SQL Anywhere コンソールからデータベースに接続します。
2. [ユーザ ID] カラムでユーザを右クリックし、[切断] を選択します。

   SQL Anywhere コンソールに表示される列は、[オプション] ダイアログで設定できます。このダイアログへは [ファイル] – [オプション] を選択するとアクセスできます。

SQL Anywhere コンソールは、Windows CE、AIX、HP-UX、HP-UX Itanium、Linux Itanium を除き、サポートされるすべてのプラットフォームで使用可能です。これらのプラットフォームでは、接続レベル、サーバレベル、データベースレベルのプロパティを使用して情報を取得したり、SQL Anywhere コンソールをサポートするオペレーティング・システム (Windows、Mac OS X、Linux など) を実行するコンピュータからサーバをモニタしたりすることができます。

プロパティ値の取得については、「データベース・プロパティ」 517 ページを参照してください。

SQL Anywhere スクリプト実行ユーティリティ (dbrunsql)

Windows CE 上で実行されているデータベースに対して、SQL コマンドやコマンド・ファイルを実行できます。

構文

dbrunsql [*options*] [*SQL-script-file* | *SQL-command*]

オプション	説明
-c *"keyword=value; ⋯"*	接続パラメータを指定します。「接続パラメータ」 230 ページを参照してください。
-d	このオプションを指定すると、結果セットからエクスポートされたデータだけが出力ファイルに書き込まれます。-d を指定しない場合は、dbrunsql のすべての出力が出力ファイルに書き込まれます。
-e [**c** | **p** | **s**]	文の実行中にエラーが発生した場合の動作を指定します。デフォルトでは、ユーザにエラーが発生したことを知らせるプロンプトを表示します。 ♦ **c** エラーを無視し、文の実行を続行します。 ♦ **p** ユーザに続行するかどうかを確認するプロンプトを表示します。 ♦ **s** 文の実行を停止します。
-f [**f** | **a**]	結果セットをエクスポートするときに使用されるデータ・フォーマットを指定します。次のいずれかの値を指定できます。 ♦ **a** データのエクスポートに ASCII フォーマットを使用します。 ♦ **f** データのエクスポートに FIXED フォーマットを使用します。これがデフォルト・フォーマットです。
-g [**+** | **-**]	デフォルトでは、dbrunsql GUI が表示されます。GUI を表示しない場合は、-g- を指定します。
-o *filename*	指定したファイルに、出力メッセージを書き込みます。
-q	出力メッセージを表示しません。これは、コマンドまたはコマンド・ファイルを使って Interactive SQL を起動したときにのみ便利です。このオプションを指定した場合、エラー・メッセージは表示されますが、次の情報は表示されません。 ♦ 警告およびその他の致命的でないメッセージ ♦ 結果セットの出力
-qc	コマンドまたはスクリプト・ファイルの実行が完了した後で dbrunsql ウィンドウを閉じる場合は、このオプションを指定します。
-s *number*	結果セットを FIXED フォーマットでエクスポートする場合は、1 つのカラムからフェッチするバイトの最大数を指定できます。デフォルト値は 255 です。

オプション	説明
-v	-v オプションを指定すると、各 SQL 文のすべての行が dbrunsql の出力に表示されます。このオプションを指定せずにスクリプト・ファイルを実行した場合は、現在実行中の行番号が表示されます。

備考

dbrunsql ユーティリティでは、データベースに対して SQL コマンドを実行したりコマンド・ファイルを実行したりすることができます。SQL Anywhere スクリプト実行ユーティリティ (dbrunsql) は Windows CE でのみサポートされます。

SQL Anywhere サポート・ユーティリティ (dbsupport)

エラーやソフトウェアの使用状況に関する情報を iAnywhere に送信します。

構文

dbsupport [*options*] *operation* [*operation-specific-option*]

dbsupport *configuration-options*

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードやその他の情報を保護する場合は、ファイル非表示ユーティリティを使用すると、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-o *filename*	指定されたファイルに出力を送信します。
-q	重要なメッセージだけを表示します。

操作	説明
-is *submission-ID* [**-rr** *N*]	-is では、iAnywhere に送信されたクラッシュ・レポートのステータスをチェックできます。 たとえば、次のコマンドは送信 ID 66 のステータスを問い合わせます。 `dbsupport -is 66`
-iu [**-r** *N*]	-iu オプションを指定することによって、使用している SQL Anywhere のビルドに対する更新のリリース状況を確認できます。 更新については、Interactive SQL と Sybase Central を使用して確認することもできます。「[オプション] ダイアログ : [更新のチェック] タブ」 『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。
-lc	-lc オプションを使用すると、iAnywhere に送信されていないクラッシュ・レポートのリストを作成できます。リストされるレポート名には、-sc オプションを使用できます。

操作	説明
-ls	-ls オプションを使用すると、iAnywhere に送信されたクラッシュ・レポートの送信 ID のリストを作成できます。次に例を示します。 dbsupport -ls このコマンドは次のような情報を返します。 Submission ID: 4 Minicore dump 20051220_133828_32116 reported: Wed Mar 15 16:31:56 2006 Submission ID: 98 Minicore dump 20051229_221211_3221 reported: Wed Mar 22 16:33:26 2006
-pc *filename*	-pc オプションを指定すると、クラッシュ・レポートの情報が出力されます。このオプションによって、情報を確認してからレポートを iAnywhere に送信できます。
-pd	-pd オプションを指定すると、収集された診断情報が出力されます。このオプションによって、情報を確認してからレポートを iAnywhere に送信できます。
-ps *submission-ID*	このオプションでは、iAnywhere に送信された特定のレポートに関する情報を表示できます。次に例を示します。 dbsupport -ps 4 このコマンドは送信 ID 4 の情報を返します。 Minicore dump 20051220_133828_32116 reported: Wed Mar 15 16:31:56 2006
-sa [**-r** *number-of-submission-retries*]	-sa オプションを指定すると、診断ディレクトリに保存されているすべてのクラッシュ・レポートと診断情報が iAnywhere に送信されます。
-sc *reportname* [**-r** *number-of-submission-retries*] [**-nr** | **-rr** *N*]	-sc オプションを指定すると、クラッシュ・レポートと診断情報が iAnywhere に送信されます。次に例を示します。 dbsupport -sc 20051220_133828_32116 -lc オプションを使用すると、送信されていないレポートのリストを参照できます。
-sd [**-r** *number-of-submission-retries*]	-sd オプションを指定すると、すべてのクラッシュ・レポートと診断情報が iAnywhere に送信されます。 診断ディレクトリの詳細については、「SADIAGDIR 環境変数」 311 ページを参照してください。

設定オプション

設定オプション	説明
-cc [**autosubmit** | **no** | **promptDefY** | **promptDefN**]	-cc オプションでは、dbsupport によるプロンプトの表示方法を変更できます。次のいずれかのオプションを指定できます。 ♦ **autosubmit** レポートを自動送信します。 ♦ **no** レポートを送信してよいかどうかを確認しません。レポートは送信されません。 ♦ **promptDefY** 可能な場合、レポートを送信してよいかどうかを確認します。回答がない場合は、レポートを送信します。 ♦ **promptDefN** 可能な場合、レポートを送信してよいかどうかを確認します。回答がない場合は、レポートを送信しません。これがデフォルトの動作です。 たとえば、アプリケーションで Embedded SQL Anywhere を使用する場合は、iAnywhere にレポートを送信しないように SQL Anywhere サポート・ユーティリティを設定することもできます。 このオプションを指定すると、その値が SQL Anywhere サポート・ユーティリティのデフォルト値となります。設定は、診断ディレクトリ内の *dbsupport.ini* ファイルに保存されます。 次のコマンドは、レポートを送信せず、ユーザに対してレポート送信の確認メッセージを表示しないように、SQL Anywhere サポート・ユーティリティを設定します。 `dbsupport -cc no`
-cd	-cd では、レポートの送信が失敗した場合にリトライするまでの遅延時間を秒単位で指定します。デフォルトの遅延時間は 30 秒です。 このオプションを指定すると、その値が SQL Anywhere サポート・ユーティリティのデフォルト値となります。設定は、診断ディレクトリ内の *dbsupport.ini* ファイルに保存されます。 次の dbsupport コマンドでは、操作が失敗した場合に 3 秒間隔で最大 4 回リトライするように指定しています。 `dbsupport -cr 4 -cd 3`

設定オプション	説明

-ch *crash-handler-program*	-ch オプションでは、クラッシュが発生したときに呼び出すプログラムを指定します。
	
	このオプションを指定すると、その値が SQL Anywhere サポート・ユーティリティのデフォルト値となります。設定は、診断ディレクトリ内の *dbsupport.ini* ファイルに保存されます。
	
	データベース・サーバは、*crash-handler-program* に渡す情報を構築する 3 つの代入パラメータをサポートしています。
	
	◆ **%F** このパラメータは、生成されたレポート・ファイルのフル・パスに置き換えられます。
	
	◆ **%P** このパラメータは、レポートを生成したプログラムの名前に置き換えられます。たとえば、バージョン 10 のパーソナル・データベース・サーバがレポートを生成した場合は、dbeng10 が返されます。
	
	◆ **%S** このパラメータは、クラッシュまたは致命的なエラーが発生したときに使用されていたデータベース・サーバの名前に置き換えられます。たとえば、Sample というデータベース・サーバがレポートを生成した場合は、Sample が返されます。
	
	%F、%P、%S の代わりに $F、$P、$S を使用できます。% と $ の解釈がコマンド・シェルによって異なるため、両方の文字を提供しています。たとえば、4NT では %F が環境変数 F の値に置き換えられるため、それを回避するために $F を使用できます。
	
	c:¥test.bat にクラッシュ・ハンドラ・プログラムがあり、次のコマンドが含まれているとします。
	
	`copy %1 c:¥archives`
	`echo %2`
	
	Windows では、クラッシュが発生した場合、次のコマンドによって dbsupport が *c:¥test.bat* を起動し、2 つのパラメータを渡します。レポートを送信する場合は、このプログラムが呼び出された後でレポートが送信されます。
	
	`dbsupport -ch "c:¥test.bat ¥"%F¥" parm2"`
	
	%F に指定されたパスが 1 つ目のパラメータとして *c:¥test.bat* に渡されます。2 つ目のパラメータとして、*parm2* が *c:¥test.bat* に渡されます。引数を取るクラッシュ・ハンドラ・プログラムを指定する場合は引用符が必要であることに注意してください。
	
	上記の例では、生成されたレポート・ファイルのフル・パスを引用符で囲んでいます。dbsupport が使用中であるためにレポート・ファイルにアクセスできなくなるという問題を回避するために、クラッシュ・ハンドラ・プログラムは上記のようにレポート・ファイルの専用コピーを作成する必要があります。

設定オプション	説明
-ch-	-ch- オプションを使用すると、*dbsupport.ini* ファイルに格納されているクラッシュ・ハンドラ設定を削除できます。次に例を示します。 dbsupport -ch-
-cid *customer-id*	-cid では、送信レポート内でユーザが自分自身を識別するための文字列を指定します。このオプションを指定すると、その値が SQL Anywhere サポート・ユーティリティのデフォルト値となります。設定は、診断ディレクトリ内の *dbsupport.ini* ファイルに保存されます。 次の例は、dbsupport に対してカスタマ ID 文字列を指定しています。 dbsupport -cid myid@company.com dbsupport -cid "MyClientApp 1.0"
-cid-	-cid- オプションを使用すると、*dbsupport.ini* ファイルからカスタマ ID 文字列を削除できます。次に例を示します。 dbsupport -cid-
-cp *email-server* [**:***port*]	HTTP プロキシのホストとポートを設定する
-cp autodetect	-cp オプションでは、クラッシュ・レポートの送信に使用する HTTP プロキシのホストとポートを設定します。 Windows では、-cp autodetect の構文もサポートされます。このオプションを指定し、Internet Explorer でプロキシ・サーバとポートが設定され、現在有効である場合、dbsupport はそのシステム設定に基づいてプロキシ・サーバとポートを設定します。Internet Explorer でプロキシ・サーバとポートを設定するには、[ツール] – [インターネット オプション] – [LAN の設定] を選択します。
-cp-	-cp- オプションを使用すると、*dbsupport.ini* ファイルから HTTP プロキシのホストとポートの設定を削除できます。次に例を示します。 dbsupport -cp-

設定オプション	説明
-cr *number-of-submission-retries*	-cr オプションでは、送信が失敗した場合のリトライ回数を指定します。 このオプションを指定すると、その値が SQL Anywhere サポート・ユーティリティのデフォルト値となります。設定は、診断ディレクトリ内の *dbsupport.ini* ファイルに保存されます。 次の dbsupport コマンドでは、操作が失敗した場合に 3 秒間隔で最大 4 回リトライするように指定しています。 dbsupport -cr 4 -cd 3

操作固有のオプション	説明
-nr	-nr を指定すると、dbsupport は送信のステータスをチェックしません。たとえば、次のコマンドは、レポートを送信するだけで、その送信のステータスについてはチェックしません。 dbsupport -nr -sc 20051220_133828_32116 デフォルトでは、送信するレポートで報告する問題について解決策がすでに提示されていないかどうかチェックします。
-r *number-of-submission-retries*	-r オプションでは、dbsupport がレポートの送信をリトライする最大回数を指定できます。0 を指定すると、リトライ回数は無限となります。デフォルト値は 10 です。*dbsupport.ini* に -cr オプションが保存されている場合、-r を指定すると -cr の値は上書きされます。
-rd *retry-delay*	-rd オプションでは、レポートを再送するまでの dbsupport の待ち時間を秒単位で指定できます。デフォルト値は 30 です。*dbsupport.ini* に -cd オプションが保存されている場合、-rd を指定すると -cd の値は上書きされます。
-rr *number-of-submission-response-retries*	-r オプションでは、dbsupport が送信に対する応答の取得をリトライする最大回数を指定できます。0 を指定すると、リトライ回数は無限となります。デフォルト値は 10 です。

備考

SQL Anywhere サポート・ユーティリティ (dbsupport) では、次のタスクを実行できます。

- 診断情報とクラッシュ・レポートをインターネット経由で iAnywhere に送信する
- 機能の統計を送信する

- 送信済みおよび未送信のクラッシュ・レポートに関する情報を表示する
- 送信済みおよび未送信のクラッシュ・レポートに関する情報を出力する
- 送信のステータスを問い合わせる
- 使用中の SQL Anywhere のビルドに対応するソフトウェア更新のリリース状況を問い合わせる
- データベース・サーバまたは Mobile Link サーバが致命的なエラー (アサーション／クラッシュ) を検出した場合の処置を設定する

致命的なエラーが発生したときに、以下のアプリケーションからエラー・レポートとして情報を送信できます。

- Interactive SQL (dbisql)
- Mobile Link Listener (dblsn)
- Mobile Link サーバ (mlsrv)
- ネットワーク・サーバ (dbsrv10)
- パーソナル・サーバ (dbeng10)
- QAnywhere Agent (qaagent)
- Replication Agent (dbltm)
- Mobile Link 用 SQL Anywhere クライアント (dbmlsync)
- SQL Anywhere コンソール・ユーティリティ (dbconsole)
- SQL Remote (dbremote)
- Sybase Central

レポートの送信が完了すると、レポートにユニークな送信 ID が割り当てられます。レポートは診断ディレクトリに書き込まれます。診断ディレクトリの詳細については、「SADIAGDIR 環境変数」 311 ページを参照してください。

エラー・レポートとその送信方法の詳細については、「SQL Anywhere のエラー・レポート」 56 ページを参照してください。

プラットフォーム	デフォルトの診断ディレクトリ
Windows (Windows CE を除く)	*%ALLUSERSPROFILE%¥Application Data¥SQL Anywhere 10¥diagnostics*
Windows CE	実行プログラムがあるディレクトリ
UNIX	*$HOME/.sqlanywhere10/diagnostics*
NetWare	*SYS:¥SYSTEM*

SQL Anywhere サポート・ユーティリティは、問題を検出したときに特定のアクションを実行するように設定できます。たとえば、データベース・サーバがエラー・レポートを送信するたびに、指定したハンドラ・プログラムを実行するように指定できます。この機能は、エラー処理プロセスにカスタム・アクションを追加する場合に便利です。

SQL Anywhere サポート・ユーティリティでは、特定の操作をリトライするように設定できます。たとえば、レポートの送信については、リトライの間隔を 30 秒、最大回数を 10 回に設定することもできます。この機能によって、サービスが一時的に使用不能となった場合の対処方法を組み込むことができます。

SQL Anywhere サポート・ユーティリティの設定は、診断ディレクトリの *dbsupport.ini* ファイルに格納されます。

操作固有のオプションは、*dbsupport.ini file* に保存されている動作を含め、デフォルトの動作に優先させる場合に役立ちます。

参照

- 「SADIAGDIR 環境変数」 311 ページ
- 「ソフトウェア・コンポーネントの終了コード」 『SQL Anywhere サーバ - プログラミング』

サーバ・バックグラウンド起動ユーティリティ (dbspawn)

バックグラウンドでデータベース・サーバを起動します。

構文

dbspawn [*options*] *server-command*

オプション	説明
@data	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。指定された環境変数と設定ファイルが両方とも存在する場合は、環境変数が使用されます。サーバ・バックグラウンド起動ユーティリティ (dbspawn) では、@data オプションで指定された設定ファイルの内容は展開されません。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードやその他の情報を保護する場合は、ファイル非表示ユーティリティを使用すると、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-f	デフォルトのデータベース・サーバがすでに存在する場合でも、dbspawn を使用してデータベース・サーバを起動するには、このオプションを使用します。データベース・サーバが実行されている場合でも、それがデフォルトのデータベース・サーバでないときは、dbspawn は別のサーバを起動します。 dbspawn が起動しようとしたデータベース・サーバと同じ名前のデータベース・サーバがすでに実行されている場合、dbspawn は新たなサーバを起動しないで成功を示す終了コードを返します。

-p	データベース・サーバ・プロセスのオペレーティング・システム・プロセス ID です。次に例を示します。
	
	`dbspawn -p dbeng10 -n newserver`
	
	次の形式のメッセージをコマンド・プロンプトにレポートします。
	
	`New process ID is 306`

-q	クワイエット・モードで実行します (メッセージを表示しません)。
server-command	データベース・サーバを開始するためのコマンド・ラインを指定します。「SQL Anywhere データベース・サーバ」 138 ページを参照してください。

備考

dbspawn ユーティリティは、サーバをバックグラウンドで起動するために用意されています。dbspawn は、バックグラウンドでサーバを起動し、終了コード 0 (成功) または 0 以外の値 (失敗)

を返します。同じコンピュータですでにデータベース・サーバが実行されている場合、dbspawn は新たなサーバを起動せず、失敗をレポートします。すでに実行されているデータベース・サーバがない場合は、データベース・サーバの初期化が完了し、要求を受信できる状態になるまで、dbspawn は終了しません。

終了コードの詳細については、「ソフトウェア・コンポーネントの終了コード」『SQL Anywhere サーバ - プログラミング』を参照してください。

dbspawn ユーティリティは、バッチ・ファイルからサーバを起動するのに役立ちます。特に、バッチ・ファイルの後続コマンドが要求を受け入れるサーバを必要とする場合に便利です。

指定するパスに 1 つ以上のスペースが含まれている場合は、パス全体を二重引用符で囲む必要があります。次に例を示します。

```
dbspawn dbeng10 "c:¥my databases¥mysalesdata.db"
```

指定したパスにスペースがない場合、引用符は必要ありません。

サーバ停止ユーティリティ (dbstop)

データベースまたはデータベース・サーバを停止します。

構文

dbstop [*options*] [*server-name*]

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードやその他の情報を保護する場合は、ファイル非表示ユーティリティを使用すると、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-c *"keyword=value; ⋯"*	ネットワーク・サーバを停止する場合は、サーバを停止するパーミッションのあるユーザ ID を接続文字列に指定します。デフォルトでは、ネットワーク・サーバに対しては DBA パーミッションが必要であり、パーソナル・サーバはすべてのユーザが停止できます。ただし、-gk サーバ・オプションを使用するとこれを変更できます。 接続パラメータを指定する場合は、サーバ名を指定しないでください。「接続パラメータ」 230 ページ、「Unconditional 接続パラメータ [UNC]」 263 ページ、「-gk サーバ・オプション」 170 ページを参照してください。
-d	データベース・サーバは停止しません。代わりに、接続文字列で指定したデータベースだけを停止します。
-o *filename*	指定したファイルに、出力メッセージを書き込みます。
-q	クワイエット・モードで実行します (メッセージを表示しません)。
-x	サーバへのアクティブな接続がある場合、サーバを停止しません。このオプションを使用すると、dbstop はアクティブな接続があるときでもプロンプトを表示しません。
-y	サーバにアクティブな接続がある場合でも、サーバを停止します。これは、接続パラメータに Unconditional=YES を含めることと同等です。

オプション	説明
server-name	現在のコンピュータ上で稼働中のデータベース・サーバの名前。シャットダウン時にパーミッションが一切必要にならないように、データベース・サーバを起動してください。パーソナル・データベース・サーバは、デフォルトではこのモードで起動します。ネットワーク・データベース・サーバの場合は、**-gk all** オプションを指定します。「-gk サーバ・オプション」 170 ページを参照してください。 サーバ名を指定する場合は、接続パラメータを指定しないでください。

備考

サーバ停止ユーティリティは、データベース・サーバを停止します。-d オプションを使用して、指定したデータベースを停止できます。

サーバ停止ユーティリティはコマンド・プロンプトでのみ実行できます。ウィンドウ・ベースの環境では、サーバ・メッセージ・ウィンドウで [シャットダウン] をクリックしてデータベース・サーバを停止できます。

サーバ停止ユーティリティ (dbstop) は NetWare では使用できません。

オプションを使用して、アクティブな接続がある場合にも接続を停止するかどうか、またサーバを停止するのかデータベースのみを停止するのかを制御できます。

サーバ上にアクティブな接続がある場合の dbstop の動作を制御できます。アクティブな接続がある場合、dbstop はそのサーバを停止するかどうかをたずねるプロンプトを表示します。この動作を変更するには、-x オプションと -y オプションを使用します。

データベース・サーバを停止できる場合、すべてのデータベースが終了し、データベース・サーバが停止して、同じ名前のサーバを起動して同じデータベースを実行できる状態になってから、dbstop が完了します。dbstop が完了しても、データベース・サーバ・プロセスは動作したままとなることがあり、そのリソース (-o サーバ・オプションで指定された出力ファイルなど) も使用中の状態が続くことがあります。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」 『SQL Anywhere サーバ - プログラミング』を参照してください。

停止ユーティリティは NetWare では使用できません。

dbstop で SQLCONNECT 環境変数を使用する場合は、-c オプションを指定する必要があります。それ以外の場合に -c オプションを使用すると、予期しない結果となることがあります。

例

myserver という名前のサーバを実行していて、データベースは起動していないとします。このサーバを停止するには、DatabaseName (DBN) 接続パラメータとしてユーティリティ・データベースを指定します。

```
dbstop -c "UID=DBA;PWD=sql;ENG=myserver;DBN=utility_db"
```

myserver という名前のサーバを実行していて、データベース *demo.db* が動作中であるとします。このサーバとデータベースを停止するには、次のように指定します。


```
dbstop -c "UID=DBA;PWD=sql;ENG=myserver"
```

myserver という名前のパーソナル・サーバを実行しているとします。接続が確立されている状態で、このサーバとデータベースを停止するには、次のように指定します。

```
dbstop -y myserver
```

myserver という名前のサーバを実行していて、データベース *demo.db* が動作中であるとします。他のデータベースやサーバを停止しないでデータベース demo.db だけを停止するには、次のコマンドを実行します。

```
dbstop -c "UID=DBA;PWD=sql;ENG=myserver;DBN=demo" -d
```

トランザクション・ログ・ユーティリティ (dblog)

データベースのトランザクション・ログを管理します。

構文

dblog [*options*] *database-file*

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-ek *key*	このオプションを使用すると、強力に暗号化されているデータベースの暗号化キーをコマンドに直接指定できます。データベースが強力に暗号化されている場合、データベースまたはトランザクション・ログを使用するには必ず暗号化キーを指定します。強力な暗号化が適用されたデータベースの場合、-ek または -ep のどちらかを指定します。両方同時には指定できません。強力に暗号化されたデータベースでは、正しいキーを指定しないとコマンドが失敗します。
-ep	このオプションを使用すると、暗号化キーの入力を求めるプロンプトを表示するよう指定できます。このオプションでは、暗号化キーを入力するためのダイアログ・ボックスが表示されます。クリア・テキストでは暗号化キーを見ることができないようにすることで、高いセキュリティが得られます。強力な暗号化が適用されたデータベースの場合、-ek または -ep のどちらかを指定します。両方同時には指定できません。強力に暗号化されたデータベースでは、正しいキーを指定しないとコマンドが失敗します。
-g *n*	このオプションは、Log Transfer Manager を使用して Replication Server をインストールするときに指定します。バックアップをリストアして世代番号を設定するときにも使用できます。このオプションは、次の Replication Server 関数と同じ関数を実行します。 `dbcc settrunc( 'ltm', 'gen_id', n )` 世代番号と dbcc の詳細については、使用している Replication Server のマニュアルを参照してください。

オプション	説明
-il	このオプションは、このデータベースでの Replication Server のインストールで Log Transfer Manager を使用することは停止したが、SQL Remote や Mobile Link 同期は引き続き使用する場合に指定します。このオプションは、delete_old_logs オプションのために保存されている Log Transfer Manager ログ・オフセットをリセットし、必要のないトランザクション・ログを削除できるようにします。 このオプションは、次の Replication Server 関数と同じ関数を実行します。 `dbcc settrunc( 'ltm', 'ignore' )` dbcc の詳細については、Replication Server のマニュアルを参照してください。
-ir	このオプションは、このデータベースで SQL Remote を使用することは停止したが、Log Transfer Manager や Mobile Link 同期は引き続き使用する場合に指定します。このオプションは、delete_old_logs オプションのために保存されている SQL Remote ログ・オフセットをリセットし、必要のないトランザクション・ログを削除できるようにします。
-is	このオプションは、このデータベースで Mobile Link 同期を使用することは停止したが、Log Transfer Manager や SQL Remot は引き続き使用する場合に指定します。このオプションは、delete_old_logs オプションのために保存されている Mobile Link ログ・オフセットをリセットし、必要のないトランザクション・ログを削除できるようにします。
-m *mirror-name*	このオプションは新しいトランザクション・ログ・ミラーのファイル名を設定します。データベースがトランザクション・ログ・ミラーを現在使っていない場合、データベースはこの設定された名前を使って起動します。すでにトランザクション・ログ・ミラーを使っている場合、データベースはトランザクション・ログ・ミラーとして新しいファイル名を使うように変更します。
-n	トランザクション・ログとミラー・ログの使用を停止します。トランザクション・ログを使用しないと、データベースはデータ・レプリケーションに参加できず、またはデータ・リカバリにトランザクション・ログを使えません。SQL Remote、Log Transfer Manager、または dbmlsync とランケーション・オフセットが存在する場合は、対応する無視オプション (Log Transfer Manager には -il、SQL Remote には -ir、dbmlsync には -is) も指定されていないかぎり、トランザクション・ログを削除することはできません。データベースで監査が有効になっている場合、トランザクション・ログの使用を停止することはできません (まず監査を無効にする必要があります)。
-o *file-name*	指定したファイルに、出力メッセージを書き込みます。

オプション	説明
-q	クワイエット・モードで実行します (メッセージを表示しません)。
-r	ミラーされたトランザクション・ログを管理しているデータベースの場合、このオプションはその動作を変更し、1 つのトランザクション・ログだけを管理するようにします。
-t *log-name*	このオプションは新しいトランザクション・ログのファイル名を設定します。データベースがトランザクション・ログを現在使っていない場合、データベースは設定されたファイル名を使って起動します。すでにトランザクション・ログを使っている場合、データベースはトランザクション・ログとして新しいファイル名を使うように変更します。
-x *n*	SQL Remote 統合データベースを再ロードするときに使用します。このオプションを使うと、トランザクション・ログの現在の相対オフセットを *n* にリセットし、データベースがレプリケーションに関わることができるようにします。「レプリケーションに参加しているデータベースのアンロードと再ロード」『SQL Remote』を参照してください。
-z *n*	SQL Remote 統合データベースを再ロードするときに使用します。このオプションを使うと、トランザクション・ログの開始オフセットを *n* にリセットし、データベースがレプリケーションに関わることができるようにします。「レプリケーションに参加しているデータベースのアンロードと再ロード」『SQL Remote』を参照してください。

備考

dblog ユーティリティで、データベースに関連するトランザクション・ログまたはトランザクション・ログ・ミラーの名前を表示、変更できます。データベースがトランザクション・ログやミラーを管理するのを停止したり、開始したりできます。

トランザクション・ログ・ミラーはトランザクション・ログの重複コピーであり、データベースによって並列に管理されています。

データベースを初期化するときに、トランザクション・ログの名前を最初に設定します。トランザクション・ログ・ユーティリティは、データベース・ファイルを処理します。トランザクション・ログ・ファイル名を変更するときは、データベース上でデータベース・サーバを実行しないでください (実行するとエラー・メッセージが表示されます)。

このユーティリティは、トランザクション・ログに関する次のような追加情報も表示します。

- バージョン番号
- トランザクション・ログ・ファイルの名前
- ミラー・ログ・ファイルの名前 (ミラー・ログがある場合)
- 現在の相対オフセット

次の方法で、トランザクション・ログ・ユーティリティにアクセスできます。

- Sybase Central の [ログ・ファイル設定の変更] ウィザードを使用する。「トランザクション・ログの場所の変更」 866 ページを参照してください。
- Interactive SQL から ALTER DATABASE *dbfile* ALTER LOG 文を使用する。「ALTER DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- コマンド・プロンプトで、dblog コマンドを入力する。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」『SQL Anywhere サーバ - プログラミング』を参照してください。

アンロード・ユーティリティ (dbunload)

SQL コマンド・ファイルにデータベースをアンロードします。

構文

dbunload [*options*] [*directory*]

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-ac *"keyword=value; ⋯"*	このオプションを使用すると、アンロード・ユーティリティは既存のデータベースに接続し、データをそのデータベースに直接再ロードするので、データベースをアンロードする操作とその結果を既存のデータベースに再ロードする操作を一度に実行することができます。このオプションは、Windows CE ではサポートされていません。 たとえば、初期化ユーティリティを使用して新しいデータベースを作成し、このオプションを使用して再ロードすることもできます。この方法は、初期化オプションを変更する場合に便利です。 次のコマンド (全体を 1 行に入力) は、*c:¥mydata.db* データベースのコピーを *c:¥mynewdata.db* という既存のデータベース・ファイルにロードします。 `dbunload -c "UID=DBA;PWD=sql;DBF=c:¥mydata.db" -ac "UID=DBA;PWD=sql;DBF=c:¥mynewdata.db"` このオプションを使用した場合、データの中間コピーはディスク上に作成されないため、コマンドにアンロード・ディレクトリは指定しません。これにより、データのセキュリティが向上します。

オプション	説明
-an *database*	このオプションを使用すると、データベースのアンロード、新規データベースの作成、データのロードを組み合わせて実行できます。このオプションは、Windows CE 上、または Intel ベースの Mac OS X でバージョン 9 以前のデータベースを再構築する場合にはサポートされません。 通常は、データベースの初期化オプションを変更しない場合に、このオプションを使用します。ソース・データベースを作成したときに指定されたオプションが、新しいデータベースの作成に使用されます。 たとえば、次のコマンド (すべて 1 行に入力) は、*mydatacopy.db* という名前の新規のデータベース・ファイルを作成し、*mydata.db* のスキーマとデータをその中にコピーします。 dbunload -c "UID=DBA;PWD=sql;DBF=c:¥mydata.db" -an c:¥mydatacopy.db このオプションを使用した場合、データの中間コピーはディスク上に作成されないため、コマンドにアンロード・ディレクトリは指定しません。これにより、データのセキュリティが向上します。 新しいデータベースが作成されると、DB 領域ファイル名に R が追加され、元のデータベースの DB 領域と同じディレクトリに新しいデータベースの DB 領域が作成された場合の名前の競合を防ぎます。たとえば、アンロードされたデータベースの *library.db* ファイルに library という DB 領域がある場合、新しいデータベースの library DB 領域は *library.dbR* となります。 -an には、ファイル名としてデータベース・サーバに対する相対パスを指定します。
-ap *size*	このオプションでは、新しいデータベースのページ・サイズを設定できます。-an または -ar を指定しない場合、このオプションは無視されます。データベースのページ・サイズには、2048、4096、8192、16384、32768 バイトのいずれかを指定できます。デフォルトは元のデータベースのページ・サイズです。このオプションとともに -an または -ar のいずれかを指定する必要があります。データベース・サーバですでにデータベースが実行中の場合、サーバのページ・サイズ (-gp オプションで設定) は新規ページ・サイズを処理するのに十分な容量でなければなりません。「-gp サーバ・オプション」 173 ページを参照してください。

オプション	説明
-ar [*directory*]	このオプションは、古いデータベースと同じ設定を持つ新しいデータベースを作成し、新しいデータベースを再ロードして、古いデータベースと置き換えます。このオプションを使用する場合は、データベースへの他の接続がなく、データベース接続が、ネットワークではなくローカルである必要があります。このオプションは、Windows CE 上、または Intel ベースの Mac OS X でバージョン 9 以前のデータベースを再構築する場合にはサポートされません。 オプションの *directory* を指定すると、レプリケーションを行うためにトランザクション・ログのオフセットがリセットされ、古いデータベースのトランザクション・ログが指定のディレクトリに移動されます。指定されたディレクトリは、Message Agent と Replication Agent が使用する古いトランザクション・ログを保持している必要があります。トランザクション・ログ管理は、データベースがレプリケーションで使用されるときだけ実行されます。SQL Remote パブリッシャや LTM チェックがない場合、古いトランザクション・ログは不要なので、指定するディレクトリにはコピーされず、削除されます。「レプリケーション・インストールにおけるリモート・データベースのバックアップ方法」 826 ページを参照してください。 新しいデータベースが作成されると、DB 領域ファイル名に R が追加され、元のデータベースの DB 領域と同じディレクトリに新しいデータベースの DB 領域が作成された場合の名前の競合を防ぎます。たとえば、アンロードされたデータベースの *library.db* ファイルに library という DB 領域がある場合、新しいデータベースの library DB 領域は *library.dbR* となります。 暗号化されたデータベースを再構築する場合は、新しいデータベースで元のデータベースと同じ暗号化キーを使用する必要があります。
-c *"keyword=value; ⋯"*	このオプションでは、ソース・データベースの接続パラメータを指定します。接続パラメータの詳細については、「接続パラメータ」 230 ページを参照してください。ユーザはデータベースの全テーブル上にパーミッションを持っている必要があるため、user ID には DBA 権限を持つユーザ ID を指定してください。 たとえば、次の文は、パスワード sql を持つユーザ ID DBA として接続し、サンプル・データベースをアンロードします。データは *c:¥unload* ディレクトリにアンロードされます。 `dbunload -c "DBF=samples-dir¥demo.db;UID=DBA;PWD=sql" c:¥unload` *samples-dir* の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。
-d	このオプションを指定すると、データベース定義コマンド (CREATE TABLE、CREATE INDEX など) は生成されません。*reload.sql* にはデータを再ロードする文のみが記述されます。

オプション	説明
-dc	このオプションを指定すると、データベース内のすべての計算カラムが再計算されます。デフォルトでは、計算カラムは再計算されません。-dc オプションを指定すると、*reload.sql* スクリプトに計算カラムの再計算に対応するセクションが追加されます。追加される文は次のような形式です。 ALTER TABLE "owner"."*table-name*" ALTER "*computed-column*" SET COMPUTE (*compute-expression*); テーブルに CURRENT DATE などのコンテキスト依存の計算カラムがある場合は、-dc オプションを使用するのではなく、ALTER TABLE 文を使用して計算カラムの値を再計算することをおすすめします。「ALTER TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
-e *table*, …	このオプションでは、指定したテーブルを *reload.sql* ファイルから除外することができます。 -e オプションを使用して作成した *reload.sql* ファイルにはすべてのデータベース・テーブルが含まれるわけではないので、そのようなファイルを使用してデータベースを再構築しないでください。外部キーが参照するテーブルがある場合、そのテーブルを除外すると、データベースを再構築することはできません。 -e オプションを使用する場合は、必ず -d オプションも使用し、-e で指定したテーブルを除くすべてのテーブルのデータをアンロードすることをおすすめします。

オプション	説明
-ea *algorithm*	新規データベースに使用する暗号化を指定します。強力な暗号化を実現するには、FIPS 認定アルゴリズム用に AES または AES_FIPS を設定し、-ek オプションを指定します。AES_FIPS では、AES とは別のライブラリが使用されます。「強力な暗号化」 936 ページを参照してください。 単純暗号化の場合は、(-ek や -ep を指定するのではなく) -ea simple を指定します。単純暗号化はデータベースの難読化に相当し、データベース・ファイルへの偶然のアクセスが発生した場合にデータが表示されないようにすることだけを目的としています。これにより第三者は、ディスク・ユーティリティを使用してファイルを表示し、データベースのデータを解読することが困難になります。セキュリティを強化するには、代わりに強力な暗号化を指定します。 暗号化されていないデータベースを作成するには、-ea オプションを指定しないか (-e、-et、-ep、-et オプションも指定しません)、-ea none を指定します。 -ea オプションを指定しない場合、デフォルトの動作は次のようになります。 ◆ -ea none、-ek (-ep、または -et が指定されない場合) ◆ -ea AES (-ek、または -ep が指定される場合) (-et は指定または指定しない) ◆ -ea simple (-ek や -ep を使用せずに -et を指定する場合) アルゴリズム名の大文字と小文字は区別されません。 -an を指定せずに -ea を指定した場合、-ea オプションは無視されます。 **別途ライセンスが必要な必須コンポーネント** ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。 「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

オプション	説明
-ek *key*	このオプションを使用すると、(-an オプションを使用して) データベースのアンロードと再ロードを実行する場合に、作成する新しいデータベースに対する dbunload コマンドに暗号化キーを指定できます。強力に暗号化されたデータベースを作成する場合、データベースやトランザクション・ログを使用するには必ず暗号化キーを指定します。データベースの暗号化に使用されるアルゴリズムは、-ea オプションで指定した AES または AES_FIPS です。-ea オプションを指定せずに、-ek オプションを指定すると、AES アルゴリズムが使用されます。-an を指定せずに -ek を指定した場合、-ek オプションは無視されます。「強力な暗号化」 936 ページを参照してください。 キーは保護してください。キーのコピーは、安全な場所に保管してください。キーを紛失すると、データベースにまったくアクセスできなくなり、リカバリも不可能になります。
-ep	このオプションを使用すると、(-an オプションを使用して) データベースのアンロードと再ロードを実行する場合に、作成する新しいデータベースに対する暗号化キーの指定を求めるプロンプトが表示されます。クリア・テキストでは暗号化キーを見ることができないようにすることで、高いセキュリティが得られます。-an を指定せずに -ep を指定した場合、-ep オプションは無視されます。-ep と -an を指定する場合は、暗号化キーが正確に入力されたことを確認するために、2 回入力してください。キーが一致しない場合は、アンロードは失敗します。「強力な暗号化」 936 ページを参照してください。
-er	-er オプションを使用すると、データベースのアンロード中に暗号化されたテーブルの暗号化を解除できます。 テーブル暗号化が有効であるデータベースを再構築するとき、-er または -et を指定して、新しいデータベースのテーブルで暗号化を有効にするかどうかを指定する必要があります。このオプションを指定しないと、データを新しいデータベースにロードしようとしたときにエラーが発生します。 次のコマンドは、テーブルの暗号化を解除してデータベース (*mydata.db*) をアンロードし、テーブル暗号化が有効になっていない新しいデータベース (*mydatacopy.db*) に再ロードします。 `dbunload -an c:¥mydatacopy.db -er -c "UID=DBA;PWD=sql;DBF=c:¥mydata.db;DBKEY=29bN8cj1z"`

オプション	説明
-et	-et オプションでは、新しいデータベースのテーブル暗号化を有効にすることができます (-an または -ar も一緒に指定する必要があります)。-ea オプションを指定せずに -et オプションを指定すると、AES アルゴリズムが使用されます。-et オプションを指定する場合、-ep または -ek も指定しないと操作は失敗します。新しいデータベースのテーブル暗号化設定を変更して、アンロードしているデータベースのテーブル暗号化設定とは異なるものに設定できます。 テーブル暗号化が有効であるデータベースを再構築するとき、-er または -et を指定して、新しいデータベースのテーブルで暗号化を有効にするかどうかを指定する必要があります。このオプションを指定しないと、データを新しいデータベースにロードしようとしたときにエラーが発生します。 次の例は、テーブルが単純暗号化アルゴリズムによって暗号化されているデータベース (*mydata.db*) をテーブル暗号化が有効になっている新しいデータベース (*mydatacopy.db*) にアンロードし、キーを 34jh として AES_FIPS アルゴリズムを使用します。 `dbunload -an c:¥mydatacopy.db -et -ea AES_FIPS -ek 34jh -c "UID=DBA;PWD=sql;DBF=c:¥mydata.db"`
-g	アンロード・ユーティリティを実行すると、dbunload は実体化ビュー (Materialized View) の定義を再作成する文を再ロード・スクリプトに追加します。デフォルトでは、これらのビューは初期化されていない状態のままとなります。つまり、データベース内でデータのないビューとして作成されます。データベースを再ロードすると、通常は実体化ビュー (Materialized View) をリフレッシュするイベントが発生し、実体化ビュー (Materialized View) にデータが配置されます。実体化ビュー (Materialized View) を再ロード・プロセス中に初期化した方がよい場合は、-g オプションを指定してください。-g を指定すると、再ロード・プロセスの最終ステップとしてシステム・プロシージャ sa_refresh_materialized_views が呼び出されます。「sa_refresh_materialized_views システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。 -g オプションを指定するかどうかを決定するときには、すべての実体化ビュー (Materialized View) を初期化すると再ロード・プロセスの実行時間が大幅に長くなる可能性があることを考慮に入れてください。一方、-g オプションを指定しない場合、初期化されていない実体化ビュー (Materialized View) を最初に使用しようとしたクエリはビューの初期化が完了するまで待つことになり、予想外の遅延が生じることがあります。-g オプションを指定しない場合は、再ロードの完了後に、実体化ビュー (Materialized View) を手動で初期化することもできます。「実体化ビュー (Materialized View) の初期化」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

オプション	説明
-ii	このオプションは、UNLOAD 文を使用してデータベースからデータを抽出し、*reload.sql* ファイル内の LOAD 文を使用してデータベースにデータを再配置します。これはデフォルトです。
-ix	このオプションは、UNLOAD 文を使用してデータベースからデータを抽出し、*reload.sql* ファイル内の Interactive SQL INPUT 文を使用してデータベースにデータを再配置します。
-k	このオプションを指定すると、sa_diagnostic_auxiliary_catalog テーブルにデータが追加されます。このテーブルは、テーブル、ユーザ、プロシージャなどのデータベース・オブジェクト ID をソース・データベースからトレーシング・データベースにマップします。このとき、すべてのヒストグラムがアンロード／再ロードされます。このオプションは、診断トレーシング情報を受け取るトレーシング・データベースを作成する場合に使用します。sa_diagnostic_auxiliary_catalog テーブルによって、サーバはトレーシング・データを収集したときの状態を (インデックス・コンサルタントやアプリケーション・プロファイリングなどの目的で) シミュレートすることができます。このオプションが特に役立つのは、-n オプションと一緒に指定した場合です。「診断トレーシングを使用した詳細なアプリケーション・プロファイリング」『SQL Anywhere サーバ - SQL の使用法』と「sa_diagnostic_auxiliary_catalog テーブル」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
-m	このオプションを指定すると、レプリケーションの対象となるデータベースでユーザ ID が保存されません。
-n	このオプションを指定すると、データベースのデータはアンロードされません。*reload.sql* には、そのデータベースの構造体だけを構築する SQL 文のみが記述されています。*reload.sql* ファイルに LOAD TABLE 文または INPUT 文を追加する場合は、-n ではなく -nl を使用してください。
-nl	このオプションの機能は -n (データなし) オプションとほぼ同じですが、このオプションによって生成される *reload.sql* ファイルには、各テーブルに対する LOAD TABLE 文または INPUT 文が含まれます。このオプションを使用した場合、ユーザ・データはアンロードされません。-nl を指定するときには、LOAD/INPUT 文を生成できるようにデータ・ディレクトリも指定する必要があります。ただし、このディレクトリにファイルは書き込まれません。このオプションを指定すると、データをアンロードしない再ロード・スクリプトを生成できます。データをアンロードするには、-d を指定します。データベースにデータのアンロードが不要なテーブルがある場合は、dbunload -d -e *table-name* と指定することによって、そのテーブルのデータをアンロードの対象から除外することができます。
-o *filename*	このオプションを使用すると、出力メッセージを指定したファイルに書き込むことができます。このファイルのロケーションは dbunload に対する相対パスで指定します。

オプション	説明
-p *char*	このオプションを使用すると、外部アンロード (dbunload -x オプション) で使用するデフォルトのエスケープ文字を別の文字に変更できます。このオプションは、このユーティリティをコマンド・プロンプトから実行する場合のみ使用できます。
-q	クワイエット・モードで実行します (メッセージまたはウィンドウを表示しません)。このオプションは、このユーティリティをコマンド・プロンプトから実行する場合のみ使用できます。-q を指定する場合、-y も指定する必要があります。-y を指定しなければ、*reload.sql* がすでに存在する場合、アンロードは失敗します。
-qc	デフォルトでは、dbunload のメッセージ・ウィンドウはユーザが閉じるまで開いたままです。アンロードの完了後にメッセージ・ウィンドウを閉じる場合は、このオプションを指定します。このオプションは Windows CE でのみ使用できます。
-r *reload-file*	このオプションを使用すると、生成される再ロード・コマンド・ファイルの名前とディレクトリを変更できます。デフォルトは現在のディレクトリの *reload.sql* です。ディレクトリは、サーバではなく、クライアント・アプリケーションの現在のディレクトリに相対します。
-t *table*,⋯	このオプションでは、アンロードするテーブルのリストを指定できます。デフォルトでは、すべてのテーブルがアンロードされます。-n オプションと一緒に使うと、テーブル定義のセットだけをアンロードできます。 -t オプションを使用して作成した *reload.sql* ファイルにはすべてのデータベース・テーブルが含まれるわけではないので、そのようなファイルを使用してデータベースを再構築しないでください。外部キーが参照するテーブルがある場合、そのテーブルを除外すると、データベースを再構築することはできません。 -t オプションを使用する場合は、必ず -d オプションも使用し、-t で指定したテーブルのデータをアンロードすることをおすすめします。
-u	通常、各テーブルでは、プライマリ・キーまたはクラスタード・インデックスが定義されている場合、それに基づいてデータが順序付けられます。矛盾したインデックスを持つデータベースをアンロードする場合に、このオプションを使用すると、矛盾したインデックスがデータの順序付けに使われることはありません。
-v	アンロード・ユーティリティは、アンロードしているテーブルの名前とアンロードが完了したローの数を表示します。このオプションは、dbunload をコマンド・プロンプトから実行する場合にのみ使用できます。

オプション	説明
-xi	このオプションは、dbunload クライアントへデータをアンロードし、生成された再ロード・コマンド・ファイル *reload.sql* 内の LOAD 文を使用してデータベースにデータを再移植することによって、外部アンロードを実行します。
-xx	このオプションは、dbunload クライアントへデータをアンロードし、生成された再ロード・コマンド・ファイル *reload.sql* 内の Interactive SQL INPUT 文を使用してデータベースにデータを再移植することによって、外部アンロードを実行します。
-y	このオプションを指定すると、確認メッセージを表示することなく、既存のコマンド・ファイルが置き換えられます。-q を指定する場合、-y も指定する必要があります。-y を指定しなければ、dbunload が既存のコマンド・ファイルを検出した場合、アンロードは失敗します。 レプリケーションに関連するデータベースをアンロードする場合、特殊な考慮事項がいくつかあります。「レプリケーションに参加しているデータベースのアンロードと再ロード」『SQL Remote』と「SQL Remote のアップグレード」『SQL Anywhere 10 - 変更点とアップグレード』を参照してください。

備考

バージョン 10 へのアップグレード
既存のデータベースをバージョン 10 データベースに再構築する方法の詳細については、「SQL Anywhere のアップグレード」『SQL Anywhere 10 - 変更点とアップグレード』を参照してください。

アンロード・ユーティリティを使ってデータベースをアンロードし、指定したディレクトリの中にデータ・ファイルのセットを入れることができます。アンロード・ユーティリティは、データベースを再構築するために、Interactive SQL コマンド・ファイルを作成します。また、各テーブルのすべてのデータをカンマ区切りの形式で、指定したディレクトリ内のファイルにアンロードします。バイナリ・データはエスケープ・シーケンスを使って正しく再現できます。

また、アンロード・ユーティリティを使って、既存のデータベースから直接新しいデータベースを作成できます。これにより、通常のディスク・ファイルに書き込まれたデータベースの内容に関するセキュリティ問題が生じる可能性を回避できます。

テーブル・データのみをアンロードする場合は、Sybase Central の [データのアンロード] ダイアログを使用すると 1 つの手順で行えます。

詳細については、「[データのアンロード] ダイアログの使用」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

レプリケーションに関連するデータベースをアンロードする場合、特殊な考慮事項がいくつかあります。「レプリケーションに参加しているデータベースのアンロードと再ロード」『SQL Remote』を参照してください。

実体化ビュー (Materialized View) が含まれるデータベース
データベースを再構築した後は、データベースで実体化ビュー (Materialized View) を再表示することをおすすめします。「実体化ビュー (Materialized View) のリフレッシュ」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

次の方法で、アンロード・ユーティリティにアクセスできます。

- Sybase Central の [データベース・アンロード] ウィザードを使用する。「[データベース・アンロード] ウィザードの使用」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- コマンド・プロンプトで、dbunload コマンドを入力する。バッチまたはコマンド・ファイルへの組み込みには、このユーティリティが便利です。

*アンロード・ユーティリティは、DBA 権限のあるユーザ ID で実行してください。*この方法でないと、すべてのデータをアンロードする権限を確実に持つことができません。また、*reload.sql* ファイルも DBA として実行する必要があります(通常、このファイルは、パスワードが sql である DBA を唯一のユーザ ID とする新しいデータベースで実行します)。

データベース・サーバの -gl オプションは、データベースからデータをアンロードするときに必要となるパーミッションを制御します。「-gl サーバ・オプション」 171 ページを参照してください。

ユーザ ID dbo は、データベース内のシステム・オブジェクトを所有します。これにはビューやストアド・プロシージャも含まれます。

アンロード・ユーティリティでは、データベース作成時に dbo ユーザ ID 用に作成されたオブジェクトをアンロードしません。データベースをアンロードすると、システム・プロシージャの再定義など、これらのオブジェクトに加えられた変更は失われます。データベースの初期化以降に dbo ユーザ ID によって作成されたオブジェクトは、アンロード・ユーティリティでアンロードされ、これらのオブジェクトは保存されます。

データベースをアンロードした場合、システム・オブジェクトに対するパーミッションの変更はアンロード*されません*。新しいデータベースでパーミッションの付与または取り消しを行う必要があります。

directory は、アンロードされたデータが置かれるディレクトリ先です。*reload.sql* コマンド・ファイルは、常にユーザの現在のディレクトリとの相対ディレクトリです。

デフォルト・モードの場合、または -ii か -ix を使用する場合、データを格納するために dbunload が使用するディレクトリは、ユーザの現在のディレクトリではなく、データベース・サーバとの相対ディレクトリになります。

-xi または -xx を使用する場合、ディレクトリはユーザの現在のディレクトリとの相対ディレクトリになります。

このモードでファイル名とパスを指定する方法の詳細については、「UNLOAD TABLE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

テーブルのリストがない場合、すべてのデータベースをアンロードします。テーブルのリストがある場合、リストにあるテーブルだけをアンロードします。

アンロードされたデータには、*reload.sql* ファイルで生成された LOAD TABLE 文のカラム・リストが含まれています。カラム・リストのアンロードにより、テーブル内のカラムを並べ替えることができます。テーブルは削除または再作成することができ、その後 *reload.sql* を使用して再移植できます。

dbunload が生成する LOAD TABLE 文は、検査制約と計算カラムを無効にします。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」『SQL Anywhere サーバ - プログラミング』を参照してください。

内部アンロードと外部アンロード、内部再ロードと外部再ロード

-ii、-ix、-xi、-xx オプションを使用して、内部アンロード、外部アンロード、内部再ロード、外部再ロードを組み合わせます。内部コマンド (UNLOAD/LOAD) を使用すると、外部コマンド (Interactive SQL の INPUT と OUTPUT 文) に比べ、パフォーマンスが大幅に向上します。ただし、内部コマンドはサーバから実行されるため、ファイルとディレクトリは、データベース・サーバに対する相対パスを使用します。外部コマンドを使用すると、ユーザの現在のディレクトリに対する相対パスを使用することになります。

Sybase Central では、サーバに相対してアンロードするか、クライアントに相対してアンロードするかを指定できます。「UNLOAD TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

データベースのアンロード、再ロード、または再構築を行うために外部アンロードや再ロードを行う場合で、データベースの文字セットと dbunload を実行しているホスト・システムの文字セットとの間に互換性がないときに、文字セットの変換を行うと、データが破損する可能性があります。これは、データベースの文字セットとホスト・システムの文字セットの間で変換を実行するときに発生します。

この問題を回避するため、データベース用の接続文字列にデータベースの文字セットを指定します (-c オプションと -ac オプションを使用)。たとえば、データベースの文字セットが UTF-8 である場合、接続文字列に "charset=utf-8" を含めます。

```
dbunload -c UID=user-ID;PWD=password;
CHARSET=utf-8;DBF=filename -ac UID=user-ID;
PWD=password;CHARSET=utf-8;ENG=server-name -xx
```

アンロードの失敗

テーブル・データの再ロードとテーブル上のインデックスの再構築が完了した後で、-ar または -an を使用してデータベースの内部再構築を行っているときに障害が発生した場合、dbunload は現在のディレクトリに *unprocessed.sql* というファイルを作成します。このファイルには、障害が発生したために実行されなかった文が記録され、障害の原因となった文もコメントとして示されています。次は、*unprocessed.sql* ファイルの例です。

```
-- The database reload failed with the following error:
-- ***** SQL error: the-SQL-ERROR
-- This script contains the statements that were not executed as a
-- result of the failure. The statement that caused the failure is
-- commented out below. To complete the reload, correct the failing
-- statement, remove the surrounding comments and execute this script.

/*
the failing statement
go
```



```
*/

setuser "DBA"
go

... the remainder of the statements to be processed
```

このファイルによって、障害の原因となった文を修正、削除、または変更し、失敗となった位置から再構築を再開できるため、再構築を初めからやり直す場合に比べて大幅な時間の節約となります。

unprocessed.sql を生成すると、dbunload は失敗を示すエラー・コードを返し、再構築が失敗したことを他のツールやスクリプトに通知します。

暗号化されたデータベース

テーブル暗号化が有効であるデータベースを再構築するとき、-er または -et を指定して、新しいデータベースのテーブルで暗号化を有効にするかどうかを指定する必要があります。このオプションを指定しないと、データを新しいデータベースにロードしようとしたときにエラーが発生します。

強力に暗号化されたデータベースをアンロードする場合は、暗号化キーを指定します。DatabaseKey (DBKEY) 接続パラメータを使用して、コマンドに暗号化キーを指定できます。または、暗号化キーを読み取り可能な文字で入力するのではなく、暗号化キーを要求するプロンプトを表示させる場合は、次に示すように -ep サーバ・オプションを使用できます。

```
dbunload -c "DBF=enc.db;START=dbeng10 -ep"
```

-an オプションを使用してデータベースをアンロードし、新しいデータベースに再ロードするときに、-ek または -ep オプションを使用して新しいデータベースに対して暗号化キーを設定する場合は、次の点を考慮してください。

- 元のデータベースが強力に暗号化されている場合は、-ek または -ep オプションではなく、-c オプションで DatabaseKey (DBKEY) 接続パラメータを使用して、元のデータベースに対するキーを指定する必要があります。
- -ek または- ep オプションを使用すると、暗号化されていないデータベースをアンロードし、強力に暗号化された新しいデータベースに再ロードできます。-ep および -an を使用するときは、キーが正しいことを確認してください。正しくない場合は、アンロードは失敗します。
- 元のデータベースが強力に暗号化されていても、-ek または -ep オプションを使用しないと、新しいデータベースは単純暗号化で暗号化されます。
- -an が指定されていない場合は、-ek と -ep オプションは無視されます。dbunload の -ek と -ep オプションが新しいデータベースに適用されるのに対し、データベース・サーバ (dbeng10/dbsrv10) オプションと DBKEY= は既存のデータベースに適用されます。
- データベースの再構築が同期またはレプリケーションを伴う場合、dbunload は、-ek または -ep オプションで指定された暗号化キーが元のデータベースおよび再構築されたデータベースの暗号化キーであると見なします。

暗号化の詳細については、「-ep サーバ・オプション」 164 ページと「DatabaseKey 接続パラメータ [DBKEY]」 242 ページを参照してください。

データベースの再構築

データベースをアンロードするときには、まずそのデータベースが動作していないことを確認します。次に、dbunload を実行します。このとき、DBA ユーザとパスワードを指定し、DBF= 接続パラメータでデータベースを指定します。

データベースを再ロードするには、新しいデータベースを作成し、作成した *reload.sql* コマンド・ファイルを Interactive SQL を通じて実行します。

アンロードと再ロードを一度に行うには、dbunload を実行するときに、上記のアンロードに必要なオプションを指定するとともに、-an オプションで新しいデータベース・ファイルの名前を指定します。 -ac と -an オプションの説明を参照してください。

アップグレード・ユーティリティ (dbupgrad)

アップグレード・ユーティリティはバージョン 10 へのアップグレードに対応していない
アップグレード・ユーティリティ (dbupgrad) では、バージョン 9.0.2 以前のデータベースをバージョン 10 にアップグレードすることはできません。以前のバージョンのデータベースをバージョン 10 にアップグレードするには、アンロードと再ロードを実行し、データベースを再構築する必要があります。「SQL Anywhere のアップグレード」 『SQL Anywhere 10 - 変更点とアップグレード』を参照してください。

jConnect サポートのインストールやデータベース内の Java のサポートの変更に加えて、システム・テーブルやビューの更新、新しいデータベース・オプションの追加、すべてのシステム・ストアド・プロシージャの再作成を実行できます。

構文

dbupgrad [*options*]

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-c *"keyword=value*; ⋯*"*	接続パラメータを指定します。「接続パラメータ」 230 ページを参照してください。 ユーザ ID には DBA 権限が必要です。 たとえば、次のコマンドは、sample10 という名前の jConnect サポートがインストールされていないデータベースに、パスワード sql を持つユーザ DBA として接続し、アップグレードします。 `dbupgrad -c "UID=DBA;PWD=sql;DBF=c:¥sa10¥sample10.db" -i`

オプション	説明
-i	Sybase jConnect JDBC ドライバを使用してシステム・カタログ情報にアクセスするには、jConnect サポートをインストールする必要があります。このオプションは、jConnect システム・オブジェクトを除外したいときに使います。その場合でも、システム情報にアクセスしないかぎり、JDBC を使用できます。必要であれば、Sybase Central または ALTER DATABASE UPGRADE 文を使用して、Sybase jConnect サポートを後から追加することもできます。「jConnect システム・オブジェクトのデータベースへのインストール」『SQL Anywhere サーバ - プログラミング』と「ALTER DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
-o *filename*	指定されたファイルに出力を書き込みます。
-q	クワイエット・モードで実行します (メッセージまたはウィンドウを表示しません)。

備考

dbupgrad ユーティリティは、このソフトウェアの以前のバージョンで作成されたデータベースをアップグレードし、現在のバージョンが提供する機能を使用できるようにします。アップグレードできる最も古いバージョンは SQL Anywhere 10.0.0 です。このソフトウェアの初期のリリースで作成したデータベースに対して、そのバージョン以降のデータベース・サーバを実行できますが、データベースを作成した後に導入された機能の中には、データベースをアップグレードしないかぎり使用できないものもあります。

実体化ビュー (Materialized View) が含まれるデータベース
データベースをアップグレードした後は、データベースで実体化ビュー (Materialized View) を再表示することをおすすめします。「実体化ビュー (Materialized View) のリフレッシュ」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

アップグレード・ユーティリティを使用すると、jConnect サポートのインストールやデータベース内の Java のサポートの変更に加えて、システム・テーブルやビューの更新、新しいデータベース・オプションの追加、データベース・オプションのリストア、すべてのシステム・ストアド・プロシージャの再作成を実行できます。

SQL Anywhere での新バージョンやソフトウェアの更新が利用可能になったため、アップグレード・ユーティリティを使って新しい機能を利用できます。

データベースのアップグレードでは、データベースをアンロードして再ロードする必要はありません。

注意
ただし、バージョン 9.0.2 以前のデータベースをバージョン 10 にアップグレードする場合は、アンロードと再ロードを実行し、データベースを再構築することが必要です。「SQL Anywhere のアップグレード」『SQL Anywhere 10 - 変更点とアップグレード』を参照してください。

アップグレードされたデータベース上でレプリケーションを使用する場合は、トランザクション・ログをアーカイブし、アップグレードされたデータベース上で新しいログを起動してください。

次の方法で、アップグレード・ユーティリティにアクセスできます。

- Sybase Central の [データベース・アップグレード] ウィザードを使用する。
- Interactive SQL の ALTER DATABASE UPGRADE 文を使用する。「ALTER DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- コマンド・プロンプトで、dbupgrad コマンドを入力する。

アップグレード前のバックアップの実行
すべてのソフトウェア・アップグレードに共通することですが、アップグレードする前にデータベースのバックアップを取ることをおすすめします。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」『SQL Anywhere サーバ - プログラミング』を参照してください。

すべての機能が使用可能になるわけではない
データベース・ファイルの物理的な再編成を必要とする機能は、dbupgrad を使用しても使用できるようにはなりません。そのような機能には、インデックスの拡張やデータの格納に関する変更が含まれています。これらの拡張機能を利用するには、データベースのアンロードと再ロードを行います。「SQL Anywhere のアップグレード」『SQL Anywhere 10 - 変更点とアップグレード』を参照してください。

検証ユーティリティ (dbvalid)

データベース内のテーブルと実体化ビュー (Materialized View) の一部またはすべてについて、インデックスとキーを検証します。

構文

dbvalid [*options*] [*object-name*, …]

オプション	説明
@*data*	このオプションを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。「設定ファイルの使用」 637 ページを参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。
-c *"keyword=value*; …*"*	接続パラメータについては、「接続パラメータ」 230 ページを参照してください。DBA 権限または VALIDATE 権限のあるユーザ ID を使用してください。 たとえば、次のコマンドは、パスワード sql を持つユーザ DBA として接続し、データベース *c:¥salesdata.db* を検証します。 dbvalid -c "UID=DBA;PWD=sql;DBF=c:¥salesdata.db"
-d	このオプションを使用すると、データベース内のすべてのテーブル・ページが正しいオブジェクトに属していることを確認し、チェックサム検証を実行することができます。-d オプションを指定した場合、データまたはインデックスは検証されません。-d オプションを -i、-s、または -t オプションと一緒に使用することはできません。
-fx	このオプションを使用すると、テーブルの各ローを検証し、ローの数がテーブルに関連付けられた各インデックスのローの数と一致することを確認できます。このオプションは、各ローに対する個々のインデックスのルックアップは実行しません。このオプションを使用すると、大規模なデータベースの検証を小さなキャッシュで行うときに、パフォーマンスを大幅に向上できます。
-i	指定のインデックスを検証します。
-o *filename*	指定したファイルに、出力メッセージを書き込みます。
-q	出力メッセージをクライアントに表示しません。ただし、-o オプションを使用してメッセージをファイルに書き込むことは可能です。

オプション	説明
-s	チェックサムは、データベース・ページがディスク上で変更されたかどうかを判断するために使用します。チェックサムを有効にしてデータベースを作成した場合、チェックサムを使用してデータベースを検証できます。チェックサム検証では、データベースの各ページをディスクから読み取って、そのチェックサムを計算します。計算したチェックサムがページに保存されているチェックサムと異なる場合、ディスク上でページが修正されていて、エラーが返されます。無効なページのページ番号がサーバ・メッセージ・ウィンドウに表示されます。-s オプションは、-d、-i、-t、またはいずれの -f オプションとも一緒に使用することはできません。
-t	*object-name* 値のリストは、テーブルと実体化ビュー (Materialized View) のリストを表します。これがデフォルトの動作です。
object-name	検証するテーブルまたは実体化ビュー (Materialized View) の名前を指定します。 -i を使用した場合、*object-name* は検証するインデックスを表します。

備考

検証ユーティリティを使用すると、データベース内のテーブルと実体化ビュー (Materialized View) の一部またはすべてについて、インデックスとキーを検証できます。検証ユーティリティでは、データベース内のすべてのテーブル・ページが正しいオブジェクトに属し、ページ・チェックサムが正しいことを確認することもできます。デフォルトでは、dbvalid はデータベース内のすべてのテーブルと実体化ビュー (Materialized View) を検証します (-t オプションを指定した場合と同じ動作です)。

検証ユーティリティは、個々のテーブルまたは実体化ビュー (Materialized View) についてオブジェクト全体をスキャンし、テーブルに定義されたインデックスとキーごとにそれぞれのレコードを調べます。検証ユーティリティでは、データベース内のすべてのテーブル・ページが正しいオブジェクトに属し、ページ・チェックサムが正しいことを確認することもできます。検証ユーティリティを実行するためには、DBA 権限または VALIDATE 権限が必要です。デフォルトでは、dbvalid はデータベース内のすべてのテーブルと実体化ビュー (Materialized View) を検証します。

検証ユーティリティを使用すると、データベース内のテーブルと実体化ビュー (Materialized View) の一部またはすべてについて、インデックスとキーを検証できます。

このユーティリティを通常のバックアップと一緒に使用すると、データベースのデータの整合性を保持できます。「バックアップとデータ・リカバリ」 811 ページを参照してください。

次の方法で、検証ユーティリティにアクセスすることもできます。

- Sybase Central の [データベース検証] ウィザードを使用する。「データベースの妥当性の確認」 828 ページを参照してください。

- Interactive SQL から VALIDATE 文を使用する。「VALIDATE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

警告
テーブルまたはデータベース全体の検証は、どの接続においてもデータベースを変更していない場合に実行してください。そうしないと、実際に破損がなくても、何らかの形でデータベースが破損したことを示す重大なエラーがレポートされます。

データベース・サーバは重要なページのチェックサムを自動的に計算するため、データベースでチェックサムが有効になっていない場合、検証ユーティリティがチェックサム違反に関する警告を返すことがあります。「データベースの妥当性の確認」 828 ページを参照してください。

そのため、検証には各テーブルに対する排他的なアクセスが必要です。このことを考慮すれば、データベース上に他のアクティビティがない場合に、データベースの検証を行うのが最適です。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。「ソフトウェア・コンポーネントの終了コード」『SQL Anywhere サーバ - プログラミング』を参照してください。

検証中に行われる個々のチェックの詳細については、「VALIDATE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

パート IV. データベースのモニタリング

パート IV では、SQL Anywhere SNMP Extension Agent の設定方法について説明します。また、SQL Anywhere SNMP Extension Agent でサポートされている OID を挙げるとともに、SQL Anywhere MIB と RDBMS MIB のテーブルの内容も示します。

第 17 章

SQL Anywhere SNMP Extension Agent

目次

SQL Anywhere SNMP Extension Agent の概要

SQL Anywhere を Windows (32 ビット・バージョン) 上で実行している場合は、SQL Anywhere SNMP Extension Agent と SNMP 管理アプリケーションを組み合わせて SQL Anywhere データベースを管理することができます。異なるコンピュータで動作している複数のデータベース・サーバ上の各データベースを 1 つのエージェントでモニタリングできます。

SQL Anywhere SNMP Extension Agent を使用すると、次の操作を実行できます。

- 個々のサーバ統計とデータベース統計の値を検索する。
- 個々のサーバ・プロパティとデータベース・プロパティの値を検索する。
- 個々の PUBLIC データベース・オプションの値を検索する。
- いずれかの PUBLIC データベース・オプションに値を設定する。
- ストアド・プロシージャを実行する。
- プロパティ値または統計値に基づいてトラップを生成する。

提供されるファイル

SQL Anywhere インストール環境には、以下の SQL Anywhere SNMP Extension Agent 用ファイルが用意されています。

- **dbsnmp10.dll**　SQL Anywhere SNMP Extension Agent このファイルは、*install-dir¥win32* にあります。
- **iAnywhere.mib**　SQL Anywhere MIB には、データベース・サーバとデータベースのプロパティ、統計、オプションのうち SQL Anywhere SNMP Extension Agent を使用してアクセスできるものの OID が、すべて格納されています。
- **RDBMS-MIB.mib**　リレーショナル・データベース管理システムのための汎用 MIB であり、SQL Anywhere SNMP Extension Agent を使用してアクセスできる OID が格納されています。
- **SNMPv2-SMI.mib**　この MIB は、Adaptive Server Anywhere MIB と RDBMS MIB から参照されます。
- **SNMPv2-TC.mib**　この MIB は、Adaptive Server Anywhere MIB と RDBMS MIB から参照されます。
- **SYBASE-MIB.mib**　Sybase MIB です。この MIB は、ASQL Anywhere MIB から参照されます。
- **sasnmp.ini**　このファイルには、SQL Anywhere SNMP Extension Agent によるモニタリングの対象となるデータベースが列記されています。デフォルトで、このファイルは *install-dir¥win32* にあります。

SNMP の概要

Simple Network Management Protocol (**SNMP**) は、ネットワーク管理に使用されている標準的なプロトコルです。SNMP では、「**マネージャ**」と「**エージェント**」が通信できます。マネージャはエージェントに要求を送り、エージェントはマネージャからのクエリに応答します。エージェントは、特定のイベントが発生した場合に「**トラップ**」と呼ばれる通知を使用してマネージャに通知することもできます。

SNMP エージェントは、管理対象オブジェクトに対する変数値の取得要求と設定要求を処理します。各変数には値が 1 つあります。この値は一般に文字列型または整数型ですが、他のデータ型の場合もあります。

変数は、1 つのグローバルな階層に保持されます。各変数には、その親の下でユニークとなる番号が割り当てられます。変数の完全名 (すべての親を含めた名前) は、「**オブジェクト識別子** (OID)」と呼ばれます。Sybase 固有の OID は、いずれも 1.3.6.1.4.1.897 から始まります。

エージェントがサポートする OID のリストは、「**Management Information Base** (MIB)」と呼ばれるファイルに保存されています。このリストには、OID の名前や型などの情報も含まれています。

MIB は、管理対象オブジェクトについてのネットワーク管理情報が格納されているデータベースです。MIB は、SQL Anywhere SNMP Extension Agent を使用してモニタリングする SQL Anywhere データベースとは別のものです。MIB オブジェクトの値は、SNMP を使用して変更または検索できます。MIB オブジェクトは 1 つの階層にまとめられており、この階層の最上位には、ネットワークについての最も一般的な情報が置かれています。SQL Anywhere SNMP Extension Agent は、以下の MIB をサポートします。

- **SQL Anywhere MIB**　SQL Anywhere SNMP Extension Agent 専用の MIB。SQL Anywhere MIB 内の OID は、いずれも 1.3.6.1.4.1.897.2 から始まります。SQL Anywhere MIB には、SQL Anywhere SNMP Extension Agent を使用して検索 (または検索と設定の両方) ができる統計、プロパティ、オプション値の OID が登録されています。「SQL Anywhere MIB」 763 ページを参照してください。
- **RDBMS MIB**　特定のベンダに依存しない、リレーショナル・データベースのための汎用 MIB。この MIB には、システム内のデータベース・サーバとデータベースについての情報が格納されています。「RDBMS MIB」 766 ページを参照してください。

SQL Anywhere MIB

SQL Anywhere MIB は、SQL Anywhere SNMP Extension Agent 用に作成されています。データベース・サーバのすべての統計とプロパティに加えて、データベースのすべての統計、プロパティ、オプションが格納されています。統計とプロパティはいくつかの例外を除いてすべて読み込み専用ですが、データベースのオプションはいずれも読み込みと書き込みの両方が可能です。

デフォルトで、SQL Anywhere MIB は *install-dir¥snmp¥iAnywhere.mib* にあります。

SQL Anywhere MIB 内のテーブルの詳細については、「SQL Anywhere MIB リファレンス」 777 ページを参照してください。

SQL Anywhere MIB 内の値を設定する方法の詳細については、「SQL Anywhere SNMP Extension Agent による値の設定」 771 ページを参照してください。

SQL Anywhere MIB の階層は、以下の説明のとおりです。

OID	名前	説明
1.3.6.1.4.897.2.1.1.*n*.*db*	saServer.saSrvStat	データベース *db* のサーバ統計 *n* の値を返す
1.3.6.1.4.897.2.1.2.*n*.*db*	saServer.saSrvProp	データベース *db* のサーバ・プロパティ *n* の値を返す
1.3.6.1.4.897.2.2.1.*n*.*db*	saDb.saDbStat	データベース *db* のデータベース統計 *n* の値を返す
1.3.6.1.4.897.2.2.2.*n*.*db*	saDb.saDbProp	データベース *db* のデータベース・プロパティ *n* の値を返す
1.3.6.1.4.897.2.2.3.*n*.*db*	saDb.saDbOpt	データベース *db* のデータベース・オプション *n* の値を返す
1.3.6.1.4.897.2.3.1	saAgent.saVersion	SQL Anywhere Extension Agent のバージョンを返す
1.3.6.1.4.897.2.3.2.*db*	saAgent.saDbConnStr	データベース *db* 用の接続文字列を返す
1.3.6.1.4.897.2.3.3.*db*	saAgent.saConnected	SQL Anywhere Extension Agent がデータベース *db* に接続されているかどうかを示す値を返す。0 を設定すると、SQL Anywhere Extension Agent はデータベースとの接続を切断します。1 を設定すると、SQL Anywhere Extension Agent はデータベースに接続します。
1.3.6.1.4.897.2.3.4.*db*	saAgent.saStarted	データベース *db* が動作しているかどうかを返す。0 を設定すると、SQL Anywhere Extension Agent はデータベースを停止します。[1] 1 を設定すると、データベースを起動しようとします。[2]

OID	名前	説明
1.3.6.1.4.897.2.3.5.*db*	saAgent.saProc	文字列 *proc_name* を設定すると、SQL Anywhere Extension Agent はデータベース内のプロシージャ *proc_name* を実行する。 proc_name('string', 4) の形式で引数を指定できます。引数を指定しなかった場合には、名前の後に空のカッコ **()** が追加されます。値の取得を行った場合には、**""** が返されます。
1.3.6.1.4.897.2.4	saMetaData	複数の仮想テーブル。各ローが SQL Anywhere MIB でサポートされる変数の 1 つを示します。

[1] この変数の値を設定することによってデータベースを停止すると、無条件で停止が実行されます。つまり、アクティブな接続がある場合でもデータベースは停止します。

[2] この変数の値を設定することによってデータベースを起動する場合は、接続文字列内に DBF パラメータを指定しておく必要があります。必要に応じて DBN と DBKEY も指定してください。また、*sasnmp.ini* ファイル内に UtilDbPwd フィールドを設定するか、サーバのデータベース起動パーミッション (-gd サーバ・オプションを使用して指定します) を all に設定することが必要です。

asaMetaData テーブル

SQL Anywhere MIB には、サポートされている変数を調べるときに SQL Anywhere Extension Agent への問い合わせの手段として利用できるメタデータ・テーブルが含まれています。

- **saSrvMetaData.saSrvStatMetaDataTable**　データベース・サーバ統計 (sa.saServer.saSrvStat に属する変数) を示します。
- **saSrvMetaData.saSrvpropMetaDataTable**　データベース・サーバ・プロパティ (sa.saServer.saSrv.Prop に属する変数) を示します。
- **saDbMetaData.saDbStatMetaDataTable**　データベース統計 (sa.saDb.saDbStat に属する変数) を示します。
- **saDbMetaData.saDbpropMetaDataTable**　データベース・プロパティ (sa.saDb.saDbProp に属する変数) を示します。
- **saDbMetaData.saDbOptMetaDataTable**　データベース・オプション (sa.saDb.saDbOpt に属する変数) を示します。

SQL Anywhere MIB メタデータ・テーブルに保存されている情報の詳細については、「asaMetaData テーブル」 778 ページを参照してください。

RDBMS MIB

RDBMS MIB は、特定のベンダに依存しない、リレーショナル・データベース管理システム製品のための汎用 MIB (RFC 1697) です。RDBMS MIB は、**仮想テーブル**を使用してサーバとデータベースについての情報を返します。ベース OID は 1.3.6.1.2.1.39 です。この MIB 内には 9 つの仮想テーブルがあります。SQL Anywhere SNMP Extension Agent は、これらの仮想テーブルのうちの 8 つをサポートします。

RDBMS MIB 内のテーブルの詳細については、「RDBMS MIB リファレンス」 801 ページを参照してください。

SQL Anywhere Extension Agent からは、RDBMS MIB 内のサポートされる変数に「*読み込み専用*」でアクセスできます。RDBMS MIB 内の変数を SQL Anywhere Extension Agent で書き込むことはできません。

仮想テーブルは、一定数の属性と、不特定の数のローで構成されています。テーブル内の要素は、GET 要求を使用して検索します。このとき、テーブルの OID にカラム番号とロー番号を追加したものを指定します。テーブル OID の後には、次のように 1 を追加する必要があります。

table.**1**.*colnum.rownum*

デフォルトで、RDBMS MIB は *install-dir¥snmp¥RDBMS-MIB.mib* にあります。

SQL Anywhere SNMP Extension Agent の使用

SQL Anywhere SNMP Extension Agent を使用するには、コンピュータに SNMP がインストールされている必要があります。また、SQL Anywhere SNMP Extension Agent を使用してモニタリングするデータベースについての情報が保存されている *sasnmp.ini* ファイルを作成する必要があります。

SNMP のインストール

SQL Anywhere Extension Agent を使用するためには、コンピュータに SNMP をインストールする必要があります。デフォルトでは、SNMP は Windows にはインストールされません。

SNMP のインストールについては、オペレーティング・システムのマニュアルを参照してください。

SNMP のインストールが完了すると、SNMP Service と SNMP Trap Service サービスが動作している状態になります。

SQL Anywhere をインストールする前に SNMP をインストールした場合は、SNMP サービスが SQL Anywhere SNMP Extension Agent を検出できるようにするため、SNMP サービスをいったん停止し、再起動する必要があります。SQL Anywhere をインストールした後で SNMP をインストールした場合は、SNMP サービスが自動的に SQL Anywhere SNMP Extension Agent を検出します。

♦ **コマンド・ラインを使用して SNMP サービスを再起動するには、次の手順に従います。**

1. コマンド・プロンプトを開き、次のコマンドを実行します。

   ```
   net stop snmp
   ```

   この操作で SNMP サービスが停止します。

2. 次のコマンドを実行します。

   ```
   net start snmp
   ```

   この操作で SNMP サービスが起動します。

SQL Anywhere SNMP Extension Agent の設定

SQL Anywhere Extension Agent では、1 つ以上のデータベースをモニタリングできます。次に示すフォーマットで *sasnmp.ini* に保存されているデータベースがモニタリングされます。

```
[SAAgent]
TrapPollTime=time-in-seconds

[DBn]
ConnStr=connection-string
UtilDbPwd=utility-database-password
CacheTime=time-in-seconds
```



```
DBSpaceCacheTime=time-in-seconds
Trapt=trap-information
Disabled=1 or 0
```

デフォルトでは、*sasnmp.ini* ファイルは SQL Anywhere のインストール時に *install-dir¥win32* ディレクトリに格納されます。

SAAgent セクション

sasnmp.ini ファイルの SAAgent セクションには、SQL Anywhere Extension Agent に関する情報があります。TrapPollTime フィールドが不要な場合は、セクション全体を省略できます。

TrapPollTime　この値は、動的トラップのポーリング頻度を指定します (動的トラップが指定されている場合)。SQL Anywhere SNMP Extension Agent は、デフォルトでは 5 秒おきに値をポーリングします。この値を 0 に設定すると、動的トラップが無効になります。このフィールドはオプションです。

DB*n* セクション

sasnmp.ini ファイルの各 **DB*n*** セクションには、特定のデータベースの情報、そのデータベースへの接続方法、そのデータベース用の動的トラップを記述します。このセクションのフィールドでは、大文字と小文字が区別されます。

n の値は、データベースを識別する番号です。この番号は、1 から開始する連番にする必要があります。不連続にすることはできません。たとえば、*sasnmp.ini* ファイルにエントリ [DB1]、[DB2]、[DB4] がある場合、エントリ [DB3] が欠如しているためエントリ [DB4] は無視されます。

ConnStr　データベースへの接続に使用される接続文字列です。データベースに接続するには、情報を不足なく指定する必要があります。このフィールドは必須です。

- ODBC データ・ソースを使用してデータベースに接続する場合、使用するデータ・ソースは *ユーザ・データ・ソース*ではなく *システム・データ・ソース*でなければなりません。
- SNMP Extention Agent はサービスとして実行されるので、統合化ログインを使用する場合は SYSTEM アカウントにマップする必要があります。しかし、この処理を行うと、サービスとして実行されるものはすべてパスワードを入力することなくデータベースに接続できてしまいます。別の方法として、サービスを実行するためのアカウントを変更し、その後でそのアカウントの統合化ログインを作成することもできます。
- 接続文字列の先頭には、文字列 ASTART=NO;IDLE=0;CON=SNMP;ASTOP=NO を付けます。この文字列に基づいて次の設定が行われます。
  - SQL Anywhere SNMP Extension Agent がデータベース・サーバを自動的に起動するのを防ぐ
  - アイドル・タイムアウトを無効にする (SQL Anywhere SNMP Extension Agent は一定時間アイドル状態になる可能性があるため)
  - 識別が可能なように接続に名前を付ける
  - SQL Anywhere SNMP Extension Agent が接続を停止するときにデータベースが停止しないようにする

sasnmp.ini ファイル内の接続文字列にこれらの値を指定すると、デフォルトの設定が無効にされ、*sasnmp.ini* に指定した値が使用されます。

UtilDbPwd　データベースを起動するように sa.agent.saStarted を設定すると、SQL Anywhere SNMP Extension Agent は、DBF パラメータ (データベース・ファイルがある場所をデータベース・サーバに知らせるためのパラメータ) を使用してデータベースに接続しようとします。ただし、データベースの起動に必要なパーミッションが DBA である場合、サーバは接続を許可しません (これはネットワーク・サーバのデフォルト設定であり、パーソナル・サーバ、ネットワーク・サーバとも -gd dba オプションを使用してこのように設定できます)。

このようなサーバ上でデータベースを起動するためには、SQL Anywhere SNMP Extension Agent が同じサーバ上ですでに動作中のデータベースに DBA 権限を持つユーザとして接続することが必要です。この条件は、ユーティリティ・データベースに接続することで満たすことができます。ユーティリティ・データベースのパスワード (-su サーバ・オプションで指定します) を *sasnmp.ini* ファイル内に指定した場合、SQL Anywhere Extension Agent は、データベースを起動するために、そのデータベースと同じサーバ上のユーティリティ・データベースに接続し、START DATABASE 文を実行してから接続を切断します。このフィールドはオプションです。

CacheTime　データベースからデータを検索するときに、検索したデータを SQL Anywhere SNMP Extension Agent 内にキャッシュすることができます。これによって、それ以後に同じタイプのデータ (サーバ・プロパティやデータベース統計など) を検索するときには、データベースとのやりとりが不要になります。データをキャッシュすると、それ以降の検索ではデータをすばやく取得できますが、そのデータは最新の状態でない可能性があります。CacheTime フィールドは、キャッシュ時間を変更したり、キャッシュを無効にしたりする (値を 0 に設定すると無効になります) ために使用できます。デフォルトでは、キャッシュ時間は 0 秒です。CacheTime パラメータを 0 に設定すると、取り出されるデータは常に最新となります。これは、要求ごとにデータがデータベースから検索されるためです。このフィールドはオプションです。

DBSpaceCacheTime　RDBMS MIB 内の rdbmsDbLimitedResourceTable には、DB 領域についての情報が格納されています。この情報も、データベースから検索されたときに SQL Anywhere Extension Agent 内にキャッシュすることができます。DB 領域情報のデフォルトのキャッシュ時間は 600 秒 (10 分) です。このフィールドは、キャッシュ時間を変更するために使用できます。値を 0 に設定すれば、キャッシュを無効にすることもできます。このフィールドはオプションです。「rdbmsDbLimitedResourceTable」 804 ページを参照してください。

Trap*t*　動的トラップを作成します。値 *t* には正の整数を指定します。指定した値は、開始値が 1 の連番になっている必要があります。不連続にすることはできません。このフィールドはオプションです。「動的トラップの作成」 773 ページを参照してください。

Disabled　SQL Anywhere SNMP Extension Agent は、このフィールドが 1 に設定されているデータベース・エントリを無視します。このことを利用すると、SQL Anywhere SNMP Extension Agent の管理対象データベースのリストから特定のデータベースを一時的に削除する場合、他のデータベースの番号を変更する必要がありません。このフィールドはオプションです。

このファイルを編集した場合は、SNMP サービスを再起動するか、SQL Anywhere SNMP Extension Agent をリセットして、新しい設定が使用されるようにする必要があります。

♦ コマンド・ラインを使用して SNMP サービスを再起動するには、次の手順に従います。

1. コマンド・プロンプトを開き、次のコマンドを実行します。


```
net stop snmp
```

この操作で SNMP サービスが停止します。

2. 次のコマンドを実行します。

```
net start snmp
```

この操作で SNMP サービスが起動します。

♦ **SQL Anywhere SNMP Extension Agent を再起動するには、次の手順に従います。**

- SNMP 管理ツールを使用して、saAgent.saRestart プロパティ (1.3.6.1.4.1.897.2.3.6) の値を **1** に変更します。

ファイル非表示ユーティリティ (dbfhide) を使用すると、単純暗号化によって *sasnmp.ini* ファイルの内容を読みにくくすることができます。「.ini ファイルの内容の非表示」 657 ページを参照してください。

サンプル sasnmp.ini ファイル

SQL Anywhere SNMP Extension Agent の *sasnmp.ini* ファイルの例を以下に示します。

```
[SAAgent]
[DB1]
ConnStr=UID=DBA;PWD=sql;ENG=server1;DBN=sales;DBF=sales.db
Trap1=1.1.5 > 50000
UtilDbPwd=test
[DB2]
ConnStr=UID=DBA;PWD=sql;ENG=server1;DBN=field;DBF=field.db
UtilDbPwd=test
Disabled=1
[DB3]
ConnStr=UID=DBA;PWD=sql;LINKS=tcpip;ENG=server2;DBN=hq;DBF=hq.db
UtilDbPwd=test
```

SAAgent セクションにパラメータが指定されていないので、SQL Anywhere SNMP Extension Agent は 5 秒おきに値をポーリングします。

SQL Anywhere SNMP Extension Agent は、2 台のサーバで動作している 3 つの異なるデータベースをモニタリングします。データベース 3 は別のコンピュータで動作しているため、プロトコルを指定する LINKS 接続パラメータが必要です。DB1 には、データベース・サーバから送信されたデータが 50000 バイトを超えている場合に起動するトラップが指定されています。

SQL Anywhere SNMP Extension Agent による値の取得

SQL Anywhere SNMP Extension Agent を使用すると、次の値を検索できます。

- データベース・サーバ・プロパティ – 「SQL Anywhere MIB サーバ・プロパティ」 783 ページを参照してください。
- データベース・サーバ統計 – 「SQL Anywhere MIB サーバ統計」 781 ページを参照してください。

- データベース・オプション – 「SQL Anywhere MIB データベース・オプション」 793 ページを参照してください。
- データベース・プロパティ – 「SQL Anywhere MIB データベース・プロパティ」 790 ページを参照してください。
- データベース統計 – 「SQL Anywhere MIB データベース統計」 787 ページを参照してください。

これらの値を取得する方法は、使用する SNMP 管理ソフトウェアによって異なります。

例

次の表は、いくつかの OID の説明と、返される値 (それぞれの OID の値) の例を示します。

OID	説明	値の例
1.3.6.1.4.1.897.2.1.1.1.1	データベース 1 のサーバ統計 ActiveReq	1
1.3.6.1.4.1.897.2.2.1.4.1	データベース 1 のデータベース統計 CacheRead	11397
1.3.6.1.4.1.897.2.2.3.5.2	データベース 2 のデータベース・オプション ansi_integer_overflow	Off
1.3.6.1.4.1.897.2.3.1	エージェントのバージョン	10.0.1(2459)
1.3.6.1.4.1.897.2.3.2.1	データベース 1 用の接続文字列	UID=DBA;PWD=sql; ENG=server1; DBN=sales; DBF=sales.db

SQL Anywhere SNMP Extension Agent による値の設定

SQL Anywhere SNMP Extension Agent は、SNMP の get クエリ、get-next クエリ、set クエリに応答します。

SQL Anywhere SNMP Extension Agent を使用すると、任意のデータベース・オプション、一部のサーバ・プロパティ、1 つのデータベース・プロパティを設定できます。

SQL Anywhere SNMP Extension Agent は、データベース・オプションの設定時に、次の文を実行します。

```
SET OPTION PUBLIC.opt = 'value'
```

データベース・プロパティとサーバ・プロパティの設定には、sa_server_option システム・プロシージャを使用します。

これらの値の設定方法は、使用する SNMP 管理ソフトウェアによって異なります。

SQL Anywhere SNMP Extension Agent を使用して設定できるオプションとプロパティの詳細については、「SQL Anywhere MIB」 778 ページを参照してください。

SQL Anywhere SNMP Extension Agent によるストアド・プロシージャの実行

SQL Anywhere MIB には、SQL Anywhere SNMP Extension Agent を使用してストアド・プロシージャを実行するための OID が格納されています。ストアド・プロシージャを実行するには、SQL Anywhere SNMP Extension Agent が接続に使用するユーザが以下に示す要件のいずれか 1 つを満たしている必要があります。

- プロシージャの EXECUTE パーミッションを持っている
- プロシージャの所有者である
- DBA 権限を持っている

プロシージャによって生成された結果セットや戻り値は無視されます。

SQL Anywhere SNMP Extension Agent を使用してストアド・プロシージャを実行するには、saAgent.saProc (OID 1.3.6.1.4.1.897.2.3.5.*db*、*db* は *sasnmp.ini* ファイル内のデータベース番号) の値としてストアド・プロシージャの名前を示す文字列を設定します。必要に応じて、プロシージャに引数を指定できます。引数を指定しなかった場合には、プロシージャ名の後に空のカッコが追加されます。

たとえば、saAgent.saProc の値として文字列 "pchin.updatesales('param1', 2)" を設定すると、ユーザ pchin が所有するストアド・プロシージャ updatesales が呼び出されます。

この OID の値としてプロシージャ名を設定する方法は、使用する SNMP 管理ソフトウェアによって異なります。「SQL Anywhere MIB」 763 ページを参照してください。

トラップの使用

トラップ は、特定のイベントが発生した場合に SNMP エージェントから送信される OID です。トラップは SNMP エージェントで開始され、SNMP 管理ソフトウェアによって検出できます。検出後、SNMP 管理ソフトウェアは、イベントを直接処理することも、SNMP エージェントに詳細を問い合わせることもできます。

トラップを受信するには、SNMP サービスを設定する必要があります。SNMP サービスはトラップ情報を受信してこれを転送しますが、デフォルトではどこにも転送されず、動作中のトラップ・リスナは何も検出しません。コンピュータにトラップが送信されるように SNMP Service を設定する方法を以下に示します。

♦ SNMP サービスを設定するには、次の手順に従います。

1. [マイ コンピュータ] を右クリックし、ポップアップ・メニューから [管理] を選択します。

   [コンピュータの管理] ダイアログが表示されます。
2. 左ウィンドウ枠で、[サービスとアプリケーション] をダブルクリックします。
3. 左ウィンドウ枠で、[サービス] をダブルクリックします。

4. 右ウィンドウ枠のサービスのリストで [SNMP Service] を見つけ、右クリックして、ポップアップ・メニューから [プロパティ] を選択します。

   SNMP Service のプロパティ・シートが表示されます。

5. [トラップ] タブをクリックします。

6. [トラップ] タブで [追加] をクリックします。

   [SNMP サービスの構成] ダイアログが表示されます。

7. [SNMP サービスの構成] ダイアログで、テキスト・ボックスに **localhost** と入力し、[追加] をクリックします。

8. [OK] をクリックして SNMP Service のプロパティ・シートを閉じます。

SQL Anywhere SNMP Extension Agent トラップ

SQL Anywhere SNMP Extension Agent は、データベース・サーバによって接続が停止されると、必ずトラップを送信します。このトラップの OID は 1.3.6.1.2.1.39.2.1 です。

データベース・ミラーリングを使用している場合に、SQL Anywhere SNMP Extension Agent とデータベース・サーバとの接続が切断されると、SQL Anywhere SNMP Extension Agent は同じデータベース・サーバへの再接続を 30 秒おきに試行します。再接続が成功しても、接続先が以前とは別のデータベース・サーバであった場合 (ServerName プロパティによって確認できます)、SQL Anywhere SNMP Extension Agent は OID 1.3.6.1.4.1.897.2.6.3 のトラップと *sasnmp.ini* ファイルから取得したデータベース ID を送信します。この場合、SQL Anywhere SNMP Extension Agent が接続していたプライマリ・データベース・サーバが停止したため、現在はミラー・サーバがプライマリ・サーバとして動作しています。「データベース・ミラーリングの概要」 888 ページを参照してください。

SQL Anywhere SNMP Extension Agent が送信する他のトラップは、すべて動的トラップです。「動的トラップの作成」 773 ページを参照してください。

動的トラップの作成

動的トラップは、特定のプロパティ、統計、またはオプションの値を含む単純な式の評価が true である場合に SQL Anywhere Extension Agent によって送信されるトラップです。動的トラップは、*sasnmp.ini* ファイル内に作成されます。*sasnmp.ini* ファイル・エントリ内のトラップ情報のフォーマットは次のとおりです。

Trap*trapnum*=[1.3.6.1.4.1.897.2.]*oid*[.*dbnum*] *op value*

trapnum 動的トラップの番号です。開始値が 1 の連番になっている必要があります。

oid プロパティ、統計、またはオプションの OID です。SQL Anywhere MIB 内または RDBMS MIB 内の OID がサポートされます。指定された OID が有効な SQL Anywhere OID または RDBMS OID ではない場合は、SQL Anywhere MIB プレフィクス 1.3.6.1.4.1.897.2. が先頭に追加されます。

SQL Anywhere MIB 内の OID の詳細については、「SQL Anywhere MIB リファレンス」 777 ページを参照してください。

RDBMS MIB 内の OID の詳細については、「RDBMS MIB リファレンス」 801 ページを参照してください。

注意
動的トラップでは、データベース・サーバまたはデータベースのプロパティ、統計、またはオプションに対応する OID のみ使用できます。

dbnum データベース番号です。このフィールドはオプションですが、指定する場合は *sasnmp.ini* ファイルの [DB*n*] セクションのデータベース番号と一致する番号を指定する必要があります。

op 次のいずれかの値を指定します。

- **=** または **==** (等しい)
- **!=**、**<>**、または **><** (等しくない)
- **<=** または **=<** (以下)
- **>=** または **=>** (以上)
- **<** (より小さい)
- **>** (より大きい)

注意
値が文字列の場合、等しいか等しくないかの評価のみがサポートされます。

value 式内で使用する値です。文字列値は、一重引用符または二重引用符で囲む必要があります。これらの引用符は値の一部として扱われません。開始引用符または終了引用符を文字列に含める必要がある場合は、それらを二重に指定する必要があります。文字列内に出現する一重引用符は二重にしないでください。

動的トラップを設定するとき、単位をキロバイト、メガバイト、ギガバイト、またはテラバイトで指定するには、それぞれ k、m、g、または t を使用します。たとえば、次のように指定することで、現在のキャッシュ・サイズが 200 MB を超えた場合にトラップがトリガするように、動的トラップを設定できます。

```
Trap1=1.3.6.1.4.1.897.2.1.1.11.1 > 200M
```

sasnmp.ini ファイル内には、任意の数の Trap フィールドを指定できます。トラップに使用される OID は 1.3.6.1.4.1.897.2.4.1 です。トラップとともに送信されるデータには次のものがあります。

- トラップ番号 (SQL Anywhere SNMP Agent によって送信される動的トラップの番号は開始値が 1 の連番)
- データベース・インデックス
- データベース名トラップ・インデックス (*sasnmp.ini* ファイル内の値)

- 変数名
- 変数値 (変数の現在値。閾値になるとはかぎらない。)

動的トラップの動作

動的トラップは、いったんトリガされると、トリガを引き起こした条件が FALSE に変わり、その後再び TRUE になるまで、再送信されません。

たとえば、1.1.11.1 >= 51200K という式を使用して動的トラップを設定した場合、サーバのキャッシュ・サイズが 50 MB (51200 KB) に達すると、トラップがトリガされます。この時点で、この動的トラップは無効になり、それ以上送信されません。このトラップが再び有効になるのは、キャッシュ・サイズがその後 50 MB 未満になった場合だけです。この後でキャッシュ・サイズが 50 MB に戻ると、そのことが通知されます。

トラップの例

トラップ情報	説明
Trap1=1.1.5 > 10000	サーバから送信されたデータが 10000 バイトを超えると、トラップが送信される。
Trap2=1.3.6.1.2.1.39.1.4.1.4.14.1 = 10485760	トランザクション・ログ・ファイルのサイズが 10 MB を超えると、トラップが送信される。

第 18 章

SQL Anywhere MIB リファレンス

目次

SQL Anywhere MIB

SNMP エージェントがサポートするオブジェクト識別子 (OID) のリストは、「Management Information Base (MIB)」と呼ばれるファイルに保存されています。このリストには、OID の名前や型などの情報も含まれています。以下の項では、SQL Anywhere SNMP Extension Agent を使用して検索や設定ができる統計、プロパティ、オプションの OID を示します。

参照

- 「SNMP の概要」 763 ページ

Agent

Agent テーブルには、SQL Anywhere SNMP Extension Agent の情報が含まれています。

書き込み可能なプロパティには、アスタリスク記号 (*) が付いています。値 *n* は、*sasnmp.ini* ファイル内のデータベース番号です。

OID	型	名前	戻り値
1.3.6.1.4.1.897.2.3.1	String	saVersion	エージェントのバージョン
1.3.6.1.4.1.897.2.3.2.*n*	String	saDBConnStr	接続文字列
1.3.6.1.4.1.897.2.3.3.*n*	Integer32	saConnected*	エージェントが接続されている場合は 1。それ以外の場合は 0。
1.3.6.1.4.1.897.2.3.4.*n*	Integer32	saStarted*	データベースが起動されている場合は 1。それ以外の場合は 0。
1.3.6.1.4.1.897.2.3.5.*n*	String	saProc*	" "
1.3.6.1.4.1.897.2.3.6	String	saRestart*	0

asaMetaData テーブル

SQL Anywhere MIB には、次のメタデータ・テーブルがあります。

- saSrvMetaData.saSrvStatMetaDataTable
- saSrvMetaData.saSrvPropMetaDataTable
- saSrvMetaData.saDbStatMetaDataTable
- saSrvMetaData.saDbPropMetaDataTable
- saSrvMetaData.saDbOptMetaDataTable

saSrvMetaData.saSrvStatMetaDataTable

このテーブルには、データベース・サーバ統計についてのメタデータが格納されています。

値 *db* は、*sasnmp.ini* ファイル内のデータベース番号です。

OID	型	名前	戻り値
1.3.6.1.4.1.897.2.4.1.1.1.1.*db*	Integer32	saSrvStatIndex	*db*
1.3.6.1.4.1.897.2.4.1.1.1.2.*db*	Integer32	saSrvStatObjType	1[1]
1.3.6.1.4.1.897.2.4.1.1.1.3.*db*	Integer32	saSrvStatType	1[2]
1.3.6.1.4.1.897.2.4.1.1.1.4.*db*	OID	saSrvStatOID	SQL Anywhere MIB エントリの OID[3]
1.3.6.1.4.1.897.2.4.1.1.1.5.*db*	String	saSrvStatName	統計名

[1] 値：1 = サーバ、2 = データベース

[2] 値：1 = 統計、2 = プロパティ、3 = オプション

[3] 返される OID にデータベース番号は含まれません。クエリ内でデータベース番号を使用するためには、あらかじめ OID にそのデータベース番号を追加しておく必要があります。

saSrvMetaData.saSrvPropMetaDataTable

このテーブルには、データベース・サーバ・プロパティについてのメタデータが格納されています。

値 *db* は、*sasnmp.ini* ファイル内のデータベース番号です。

OID	型	名前	戻り値
1.3.6.1.4.1.897.2.4.1.2.1.1.*db*	Integer32	saSrvPropIndex	*db*
1.3.6.1.4.1.897.2.4.1.2.1.2.*db*	Integer32	saSrvPropObjType	1[1]
1.3.6.1.4.1.897.2.4.1.2.1.3.*db*	Integer32	saSrvPropType	2[2]
1.3.6.1.4.1.897.2.4.1.2.1.4.*db*	OID	saSrvPropOID	SQL Anywhere MIB エントリの OID[3]
1.3.6.1.4.1.897.2.4.1.2.1.5.*db*	String	saSrvPropName	プロパティ名

[1] 値：1 = サーバ、2 = データベース

[2] 値：1 = 統計、2 = プロパティ、3 = オプション

[3] 返される OID にデータベース番号は含まれません。クエリ内でデータベース番号を使用するためには、あらかじめ OID にそのデータベース番号を追加しておく必要があります。

saDbMetaData.saDbStatMetaDataTable

このテーブルには、データベース統計についてのメタデータが格納されています。

値 *db* は、*sasnmp.ini* ファイル内のデータベース番号です。

OID	型	名前	戻り値
1.3.6.1.4.1.897.2.4.2.1.1.1.*db*	Integer32	saDbStatIndex	*db*
1.3.6.1.4.1.897.2.4.2.1.1.2.*db*	Integer32	saDbStatObjType	2[1]
1.3.6.1.4.1.897.2.4.2.1.1.3.*db*	Integer32	saDbStatType	1[2]
1.3.6.1.4.1.897.2.4.2.1.1.4.*db*	OID	saDbStatOID	SQL Anywhere MIB エントリの OID[3]
1.3.6.1.4.1.897.2.4.2.1.1.5.*db*	String	saDbStatName	統計名

[1] 値：1 = サーバ、2 = データベース

[2] 値：1 = 統計、2 = プロパティ、3 = オプション

[3] 返される OID にデータベース番号は含まれません。クエリ内でデータベース番号を使用するためには、あらかじめ OID にそのデータベース番号を追加しておく必要があります。

saDbMetaData.saDbPropMetaDataTable

このテーブルには、データベース・プロパティについてのメタデータが格納されています。

値 *db* は、*sasnmp.ini* ファイル内のデータベース番号です。

OID	型	名前	戻り値
1.3.6.1.4.1.897.2.4.2.2.1.1.*db*	Integer32	saDbPropIndex	*db*
1.3.6.1.4.1.897.2.4.2.2.1.2.*db*	Integer32	saDbPropObjType	2[1]
1.3.6.1.4.1.897.2.4.2.2.1.3.*db*	Integer32	saDbPropType	2[2]
1.3.6.1.4.1.897.2.4.2.2.1.4.*db*	OID	saDbPropOID	SQL Anywhere MIB エントリの OID[3]
1.3.6.1.4.1.897.2.4.2.2.1.5.*db*	String	saDbPropName	プロパティ名

[1] 値：1 = サーバ、2 = データベース

[2] 値：1 = 統計、2 = プロパティ、3 = オプション

[3] 返される OID にデータベース番号は含まれません。クエリ内でデータベース番号を使用するためには、あらかじめ OID にそのデータベース番号を追加しておく必要があります。

saDbMetaData.saDbOptMetaDataTable

このテーブルには、データベース・オプションについてのメタデータが格納されています。

値 *db* は、*sasnmp.ini* ファイル内のデータベース番号です。

OID	型	名前	戻り値
1.3.6.1.4.1.897.2.4.2.1.1.1.*db*	Integer32	saDbOptIndex	*db*
1.3.6.1.4.1.897.2.4.2.1.1.2.*db*	Integer32	saDbOptObjType	2[1]
1.3.6.1.4.1.897.2.4.2.1.1.3.*db*	Integer32	saDbOptType	3[2]
1.3.6.1.4.1.897.2.4.2.1.1.4.*db*	OID	saDbOptOID	SQL Anywhere MIB エントリの OID[3]
1.3.6.1.4.1.897.2.4.2.1.1.5.*db*	String	saDbOptName	オプション名

[1] 値：1 = サーバ、2 = データベース

[2] 値：1 = 統計、2 = プロパティ、3 = オプション

[3] 返される OID にデータベース番号は含まれません。クエリ内でデータベース番号を使用するためには、あらかじめ OID にそのデータベース番号を追加しておく必要があります。

SQL Anywhere MIB サーバ統計

次の表は、SQL Anywhere SNMP Extension Agent を使用して検索できるデータベース・サーバ統計の OID と名前を示します。

値 *n* は、*sasnmp.ini* ファイル内のデータベース番号です。

データベース・サーバ統計の詳細については、「サーバ・レベルのプロパティ」 540 ページと「データベース・レベルのプロパティ」 550 ページを参照してください。

OID	型	名前	統計情報
1.3.6.1.4.1.897.2.1.1.1.*n*	Integer32	srvStatActiveReq	ActiveReq
1.3.6.1.4.1.897.2.1.1.2.*n*	Integer32	srvStatAvailIO	AvailIO
1.3.6.1.4.1.897.2.1.1.3.*n*	Counter64	srvStatBytesReceived	BytesReceived
1.3.6.1.4.1.897.2.1.1.4.*n*	Counter64	srvStatBytesReceivedUncomp	BytesReceivedUncomp
1.3.6.1.4.1.897.2.1.1.5.*n*	Counter64	srvStatBytesSent	BytesSent
1.3.6.1.4.1.897.2.1.1.6.*n*	Counter64	srvStatBytesSentUncomp	BytesSentUncomp
1.3.6.1.4.1.897.2.1.1.7.*n*	Counter64	srvStatCacheHitsEng	CacheHitsEng

OID	型	名前	統計情報
1.3.6.1.4.1.897.2.1.1.8.*n*	Integer32	srvStatCachePinned	CachePinned
1.3.6.1.4.1.897.2.1.1.9.*n*	Counter64	srvStatCacheReadEng	CacheReadEng
1.3.6.1.4.1.897.2.1.1.10.*n*	Counter64	srvStatCacheReplacements	CacheReplacements
1.3.6.1.4.1.897.2.1.1.11.*n*	Integer32	srvStatCurrentCacheSize	CurrentCacheSize
1.3.6.1.4.1.897.2.1.1.12.*n*	Counter64	srvStatDiskReadEng	DiskReadEng
1.3.6.1.4.1.897.2.1.1.13.*n*	Integer32	srvStatFreeBuffers	FreeBuffers
1.3.6.1.4.1.897.2.1.1.14.*n*	Integer32	srvStatInternal	Internal
1.3.6.1.4.1.897.2.1.1.15.*n*	Integer32	srvStatUniqueClientAddresses	UniqueClientAddresses
1.3.6.1.4.1.897.2.1.1.16.*n*	Integer32	srvStatLockedHeapPages	LockedHeapPages
1.3.6.1.4.1.897.2.1.1.17.*n*	Counter64	srvStatMainHeapBytes	MainHeapBytes
1.3.6.1.4.1.897.2.1.1.18.*n*	Integer32	srvStatMainHeapPages	MainHeapPages
1.3.6.1.4.1.897.2.1.1.19.*n*	Integer32	srvStatMapPhysicalMemoryEng	MapPhysicalMemoryEng
1.3.6.1.4.1.897.2.1.1.20.*n*	Integer32	srvStatMaxCacheSize	MaxCacheSize
1.3.6.1.4.1.897.2.1.1.21.*n*	Integer32	srvStatMinCacheSize	MinCacheSize
1.3.6.1.4.1.897.2.1.1.22.*n*	Counter64	srvStatMultiPacketsReceived	MultiPacketsReceived
1.3.6.1.4.1.897.2.1.1.23.*n*	Counter64	srvStatMultiPacketsSent	MultiPacketsSent
1.3.6.1.4.1.897.2.1.1.24.*n*	Counter64	srvStatPacketsReceived	PacketsReceived
1.3.6.1.4.1.897.2.1.1.25.*n*	Counter64	srvStatPacketsReceivedUncomp	PacketsReceivedUncomp
1.3.6.1.4.1.897.2.1.1.26.*n*	Counter64	srvStatPacketsSent	PacketsSent
1.3.6.1.4.1.897.2.1.1.27.*n*	Counter64	srvStatPacketsSentUncomp	PacketsSentUncomp
1.3.6.1.4.1.897.2.1.1.28.*n*	Integer32	srvStatPeakCacheSize	PeakCacheSize
1.3.6.1.4.1.897.2.1.1.29.*n*	Integer32	srvStatRemoteputWait	RemoteputWait
1.3.6.1.4.1.897.2.1.1.30.*n*	Integer32	srvStatReq	Req
1.3.6.1.4.1.897.2.1.1.31.*n*	Counter64	srvStatSendFail	SendFail
1.3.6.1.4.1.897.2.1.1.32.*n*	Integer32	srvStatTotalBuffers	TotalBuffers

OID	型	名前	統計情報
1.3.6.1.4.1.897.2.1.1.33.*n*	Integer32	srvStatUnschReq	UnschReq
1.3.6.1.4.1.897.2.1.1.34.*n*	Integer32	srvStatInternal	Internal
1.3.6.1.4.1.897.2.1.1.35.*n*	Integer32	srvStatCacheFile	CacheFile
1.3.6.1.4.1.897.2.1.1.36.*n*	Integer32	srvStatCacheFileDirty	CacheFileDirty
1.3.6.1.4.1.897.2.1.1.37.*n*	Integer32	srvStatCacheAllocated	CacheAllocated
1.3.6.1.4.1.897.2.1.1.38.*n*	Integer32	srvStatCachePanics	CachePanics
1.3.6.1.4.1.897.2.1.1.39.*n*	Integer32	srvStatCacheFree	CacheFree
1.3.6.1.4.1.897.2.1.1.40.*n*	Integer32	srvStatCacheScavenges	CacheScavenges
1.3.6.1.4.1.897.2.1.1.41.*n*	Integer32	srvStatCacheScavengeVisited	CacheScavengeVisited
1.3.6.1.4.1.897.2.1.1.42.*n*	Integer32	srvStatLockedCursorPages	LockedCursorPages
1.3.6.1.4.1.897.2.1.1.43.*n*	Integer32	srvStatQueryHeapPages	QueryHeapPages
1.3.6.1.4.1.897.2.1.1.44.*n*	Integer32	srvStatCarverHeapPages	CarverHeapPages
1.3.6.1.4.1.897.2.1.1.45.*n*	Integer32	srvStatHeapsRelocatable	HeapsRelocatable
1.3.6.1.4.1.897.2.1.1.46.*n*	Integer32	srvStatHeapsLocked	HeapsLocked
1.3.6.1.4.1.897.2.1.1.47.*n*	Integer32	srvStatHeapsQuery	HeapsQuery
1.3.6.1.4.1.897.2.1.1.48.*n*	Integer32	srvStatHeapsCarver	HeapsCarver
1.3.6.1.4.1.897.2.1.1.49.*n*	Integer32	srvStatMultiPageAllocs	MultiPageAllocs
1.3.6.1.4.1.897.2.1.1.50.*n*	Integer32	srvStatRequestsReceived	RequestsReceived
1.3.6.1.4.1.897.2.1.1.51.*n*	Integer32	srvStatExchangeTasks	ExchangeTasks
1.3.6.1.4.1.897.2.1.1.52.*n*	Integer32	srvStatClientStmtCacheHits	ClientStmtCacheHits
1.3.6.1.4.1.897.2.1.1.53.*n*	Integer32	srvStatClientStmtCacheMisses	ClientStmtCacheMisses

SQL Anywhere MIB サーバ・プロパティ

次の表は、SQL Anywhere SNMP Extension Agent を使用して検索できるデータベース・サーバ・プロパティの OID と名前を示します。

書き込み可能なプロパティには、アスタリスク記号 (*) が付いています。値 *n* は、*sasnmp.ini* ファイル内のデータベース番号です。

データベース・サーバ・プロパティの詳細については、「データベース・レベルのプロパティ」 550 ページを参照してください。

OID	型	名前	プロパティ
1.3.6.1.4.1.897.2.1.2.1.*n*	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.2.*n*	String	srvPropCharSet	CharSet
1.3.6.1.4.1.897.2.1.2.3.*n*	String	srvPropCommandLine	CommandLine
1.3.6.1.4.1.897.2.1.2.4.*n*	String	srvPropCompactPlatformVer	CompactPlatformVer
1.3.6.1.4.1.897.2.1.2.5.*n*	String	srvPropCompanyName	CompanyName
1.3.6.1.4.1.897.2.1.2.6.*n*	String	srvPropConnsDisabled*	ConnsDisabled
1.3.6.1.4.1.897.2.1.2.7.*n*	String	srvPropConsoleLogFile	ConsoleLogFile
1.3.6.1.4.1.897.2.1.2.8.*n*	String	srvPropDefaultCollation	DefaultCollation
1.3.6.1.4.1.897.2.1.2.9.*n*	String	srvPropIdleTimeout	IdleTimeout
1.3.6.1.4.1.897.2.1.2.10.*n*	String	srvPropIsIQ	IsIQ
1.3.6.1.4.1.897.2.1.2.11.*n*	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.12.*n*	String	srvPropIsNetworkServer	IsNetworkServer
1.3.6.1.4.1.897.2.1.2.13.*n*	String	srvPropIsRuntimeServer	IsRuntimeServer
1.3.6.1.4.1.897.2.1.2.14.*n*	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.15.*n*	String	srvPropLanguage	Language
1.3.6.1.4.1.897.2.1.2.16.*n*	String	srvPropLegalCopyright	LegalCopyright
1.3.6.1.4.1.897.2.1.2.17.*n*	String	srvPropLegalTrademarks	LegalTrademarks
1.3.6.1.4.1.897.2.1.2.18.*n*	String	srvPropLicenseCount	LicenseCount
1.3.6.1.4.1.897.2.1.2.19.*n*	String	srvPropLicensedCompany	LicensedCompany
1.3.6.1.4.1.897.2.1.2.20.*n*	String	srvPropLicensedUser	LicensedUser
1.3.6.1.4.1.897.2.1.2.21.*n*	String	srvPropLicenseType	LicenseType
1.3.6.1.4.1.897.2.1.2.22.*n*	String	srvPropLivenessTimeout*	LivenessTimeout
1.3.6.1.4.1.897.2.1.2.23.*n*	String	srvPropMachineName	MachineName
1.3.6.1.4.1.897.2.1.2.24.*n*	String	srvPropMaxMessage	MaxMessage
1.3.6.1.4.1.897.2.1.2.25.*n*	String	srvPropMessageWindowSize	MessageWindowSize

OID	型	名前	プロパティ
1.3.6.1.4.1.897.2.1.2.26.*n*	String	srvPropName	Name
1.3.6.1.4.1.897.2.1.2.27.*n*	String	srvPropNativeProcessorArchitecture	NativeProcessorArchitecture
1.3.6.1.4.1.897.2.1.2.28.*n*	String	srvPropNumPhysicalProcessors	NumPhysicalProcessors
1.3.6.1.4.1.897.2.1.2.29.*n*	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.30.*n*	String	srvPropOmniIdentifier	OmniIdentifier
1.3.6.1.4.1.897.2.1.2.31.*n*	String	srvPropPageSize	PageSize
1.3.6.1.4.1.897.2.1.2.32.*n*	String	srvPropPlatform	Platform
1.3.6.1.4.1.897.2.1.2.33.*n*	String	srvPropPlatformVer	PlatformVer
1.3.6.1.4.1.897.2.1.2.34.*n*	String	srvPropProcessCPU	ProcessCPU
1.3.6.1.4.1.897.2.1.2.35.*n*	String	srvPropProcessCPUSystem	ProcessCPUSystem
1.3.6.1.4.1.897.2.1.2.36.*n*	String	srvPropProcessCPUUser	ProcessCPUUser
1.3.6.1.4.1.897.2.1.2.37.*n*	String	srvPropProcessorArchitecture	ProcessorArchitecture
1.3.6.1.4.1.897.2.1.2.38.*n*	String	srvPropProductName	ProductName
1.3.6.1.4.1.897.2.1.2.39.*n*	String	srvPropProductVersion	ProductVersion
1.3.6.1.4.1.897.2.1.2.40.*n*	String	srvPropQuittingTime*	QuittingTime
1.3.6.1.4.1.897.2.1.2.41.*n*	String	srvPropRememberLastStatement*	RememberLastStatement
1.3.6.1.4.1.897.2.1.2.42.*n*	String	srvPropRequestFilterConn	RequestFilterConn
1.3.6.1.4.1.897.2.1.2.43.*n*	String	srvPropRequestFilterDB	RequestFilterDB
1.3.6.1.4.1.897.2.1.2.44.*n*	String	srvPropRequestLogFile*	RequestLogFile
1.3.6.1.4.1.897.2.1.2.45.*n*	String	srvPropRequestLogging*	RequestLogging
1.3.6.1.4.1.897.2.1.2.46.*n*	String	srvPropRequestLogMaxSize	RequestLogMaxSize
1.3.6.1.4.1.897.2.1.2.47.*n*	String	srvPropStartTime	StartTime
1.3.6.1.4.1.897.2.1.2.48.*n*	String	srvPropTempDir	TempDir
1.3.6.1.4.1.897.2.1.2.49.*n*	String	srvPropMultiProgrammingLevel	MultiProgrammingLevel
1.3.6.1.4.1.897.2.1.2.50.*n*	String	srvPropTimeZoneAdjustment	TimeZoneAdjustment

OID	型	名前	プロパティ
1.3.6.1.4.1.897.2.1.2.51.*n*	String	srvPropHttpPorts	HttpPorts
1.3.6.1.4.1.897.2.1.2.52.*n*	String	srvPropHttpsPorts	HttpsPorts
1.3.6.1.4.1.897.2.1.2.53.*n*	String	srvPropProfileFilterConn	ProfileFilterConn
1.3.6.1.4.1.897.2.1.2.54.*n*	String	srvPropProfileFilterUser	ProfileFilterUser
1.3.6.1.4.1.897.2.1.2.55.*n*	String	srvPropRequestLogNumFiles	RequestLogNumFIles
1.3.6.1.4.1.897.2.1.2.56.*n*	String	srvPropIsFipsAvailable	IsFipsAvailable
1.3.6.1.4.1.897.2.1.2.57.*n*	String	srvPropFipsMode	FipsMode
1.3.6.1.4.1.897.2.1.2.58.*n*	String	srvPropStartDBPermission	StartDBPermission
1.3.6.1.4.1.897.2.1.2.59.*n*	String	srvPropServerName	ServerName
1.3.6.1.4.1.897.2.1.2.60.*n*	String	srvPropRememberLastPlan	RememberLastPlan
1.3.6.1.4.1.897.2.1.2.61.*n*	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.62.*n*	String	srvPropInternal	Internal
1.3.6.1.4.1.897.2.1.2.63.*n*	String	srvPropRequestTiming	RequestTiming
1.3.6.1.4.1.897.2.1.2.64.*n*	String	srvPropCacheSizingStatistics	CacheSizingStatistics
1.3.6.1.4.1.897.2.1.2.65.*n*	String	srvPropConsoleLogMaxSize	ConsoleLogMaxSize
1.3.6.1.4.1.897.2.1.2.66.*n*	String	srvPropDebuggingInformation	DebuggingInformation
1.3.6.1.4.1.897.2.1.2.67.*n*	String	srvPropMessage	Message
1.3.6.1.4.1.897.2.1.2.68.*n*	String	srvPropMessageText	MessageText
1.3.6.1.4.1.897.2.1.2.69.*n*	String	srvPropMessageTime	MessageTime
1.3.6.1.4.1.897.2.1.2.70.*n*	String	srvPropIsRsaAvailable	IsRsaAvailable
1.3.6.1.4.1.897.2.1.2.71.*n*	String	srvPropIsEccAvailable	IsEccAvailable
1.3.6.1.4.1.897.2.1.2.72.*n*	String	srvPropMaxConnections	MaxConnections
1.3.6.1.4.1.897.2.1.2.73.*n*	String	srvPropNumLogicalProcessors	NumLogicalProcessors
1.3.6.1.4.1.897.2.1.2.74.*n*	String	srvPropNumLogicalProcessors Used	NumLogicalProcessorsUsed
1.3.6.1.4.1.897.2.1.2.75.*n*	String	srvPropNumPhysicalProcessors Used	NumPhysicalProcessorsUsed
1.3.6.1.4.1.897.2.1.2.76.*n*	String	srvPropDefaultNcharCollation	DefaultNcharCollation

OID	型	名前	プロパティ
1.3.6.1.4.1.897.2.1.2.77.*n*	String	srvPropCollectStatistics	CollectStatistics
1.3.6.1.4.1.897.2.1.2.78.*n*	String	srvPropFirstOption	FirstOption
1.3.6.1.4.1.897.2.1.2.79.*n*	String	srvPropLastOption	LastOption
1.3.6.1.4.1.897.2.1.2.80.*n*	String	srvPropLastConnectionProperty	LastConnectionProperty
1.3.6.1.4.1.897.2.1.2.81.*n*	String	srvPropLastDatabaseProperty	LastDatabaseProperty
1.3.6.1.4.1.897.2.1.2.82.*n*	String	srvPropLastServerProperty	LastServerProperty

SQL Anywhere MIB データベース統計

次の表は、SQL Anywhere SNMP Extension Agent を使用して検索できるデータベース統計の OID と名前を示します。

値 *n* は、*sasnmp.ini* ファイル内のデータベース番号です。

データベース統計の詳細については、「サーバ・レベルのプロパティ」 540 ページと「データベース・レベルのプロパティ」 550 ページを参照してください。

OID	型	名前	プロパティ
1.3.6.1.4.1.897.2.2.1.1.*n*	Counter64	dbStatCacheHits	CacheHits
1.3.6.1.4.1.897.2.2.1.2.*n*	Integer32	dbStatCacheReadIndInt	CacheReadIndInt
1.3.6.1.4.1.897.2.2.1.3.*n*	Integer32	dbStatCacheReadIndLeaf	CacheReadIndLeaf
1.3.6.1.4.1.897.2.2.1.4.*n*	Counter64	dbStatCacheRead	CacheRead
1.3.6.1.4.1.897.2.2.1.5.*n*	Integer32	dbStatCacheReadTable	CacheReadTable
1.3.6.1.4.1.897.2.2.1.6.*n*	Integer32	dbStatChkpt	Chkpt
1.3.6.1.4.1.897.2.2.1.7.*n*	Integer32	dbStatChkptFlush	ChkptFlush
1.3.6.1.4.1.897.2.2.1.8.*n*	Integer32	dbStatChkptPage	ChkptPage
1.3.6.1.4.1.897.2.2.1.9.*n*	Integer32	dbStatCheckpointUrgency	CheckpointUrgency
1.3.6.1.4.1.897.2.2.1.10.*n*	Integer32	dbStatCheckpointLogBitmapSize	CheckpointLogBitmapSize
1.3.6.1.4.1.897.2.2.1.11.*n*	Integer32	dbStatCheckpointLogBitmapPagesWritten	CheckpointLogBitmapPagesWritten

OID	型	名前	プロパティ
1.3.6.1.4.1.897.2.2.1.12.*n*	Integer32	dbStatCheckpointLogCommitToDisk	CheckpointLogCommitToDisk
1.3.6.1.4.1.897.2.2.1.13.*n*	Integer32	dbStatCheckpointLogPageInUse	CheckpointLogPageInUse
1.3.6.1.4.1.897.2.2.1.14.*n*	Integer32	dbStatCheckpointLogPagesRelocated	CheckpointLogPagesRelocated
1.3.6.1.4.1.897.2.2.1.15.*n*	Integer32	dbStatCheckpointLogSavePreimage	CheckpointLogSavePreimage
1.3.6.1.4.1.897.2.2.1.16.*n*	Integer32	dbStatCheckpointLogSize	CheckpointLogSize
1.3.6.1.4.1.897.2.2.1.17.*n*	Integer32	dbStatCheckpointLogWrites	CheckpointLogWrites
1.3.6.1.4.1.897.2.2.1.18.*n*	Integer32	dbStatCheckpointLogPagesWritten	CheckpointLogPagesWritten
1.3.6.1.4.1.897.2.2.1.19.*n*	Integer32	dbStatCommitFile	CommitFile
1.3.6.1.4.1.897.2.2.1.20.*n*	Integer32	dbStatConnCount	ConnCount
1.3.6.1.4.1.897.2.2.1.21.*n*	Integer32	dbStatCurrIO	CurrIO
1.3.6.1.4.1.897.2.2.1.22.*n*	Integer32	dbStatCurrRead	CurrRead
1.3.6.1.4.1.897.2.2.1.23.*n*	Integer32	dbStatCurrWrite	CurrWrite
1.3.6.1.4.1.897.2.2.1.24.*n*	Integer32	dbStatDiskReadIndInt	DiskReadIndInt
1.3.6.1.4.1.897.2.2.1.25.*n*	Integer32	dbStatDiskReadIndLeaf	DiskReadIndLeaf
1.3.6.1.4.1.897.2.2.1.26.*n*	Counter64	dbStatDiskRead	DiskRead
1.3.6.1.4.1.897.2.2.1.27.*n*	Integer32	dbStatDiskReadTable	DiskReadTable
1.3.6.1.4.1.897.2.2.1.28.*n*	Counter64	dbStatDiskWrite	DiskWrite
1.3.6.1.4.1.897.2.2.1.29.*n*	Integer32	dbStatExtendDB	ExtendDB

OID	型	名前	プロパティ
1.3.6.1.4.1.897.2.2.1.30.*n*	Integer32	dbStatExtendTempWrite	ExtendTempWrite
1.3.6.1.4.1.897.2.2.1.31.*n*	Integer32	dbStatFullCompare	FullCompare
1.3.6.1.4.1.897.2.2.1.32.*n*	Integer32	dbStatGetData	GetData
1.3.6.1.4.1.897.2.2.1.33.*n*	Integer32	dbStatIdleCheck	IdleCheck
1.3.6.1.4.1.897.2.2.1.34.*n*	Integer32	dbStatIdleChkpt	IdleChkpt
1.3.6.1.4.1.897.2.2.1.35.*n*	Integer32	dbStatIdleChkTime	IdleChkTime
1.3.6.1.4.1.897.2.2.1.36.*n*	Integer32	dbStatIdleWrite	IdleWrite
1.3.6.1.4.1.897.2.2.1.37.*n*	Integer32	dbStatIndAdd	IndAdd
1.3.6.1.4.1.897.2.2.1.38.*n*	Integer32	dbStatIndLookup	IndLookup
1.3.6.1.4.1.897.2.2.1.39.*n*	Integer32	dbStatIOToRecover	IOToRecover
1.3.6.1.4.1.897.2.2.1.40.*n*	Integer32	dbStatInternal	Internal
1.3.6.1.4.1.897.2.2.1.41.*n*	Integer32	dbStatInternal	Internal
1.3.6.1.4.1.897.2.2.1.42.*n*	Integer32	dbStatLockTablePages	LockTablePages
1.3.6.1.4.1.897.2.2.1.43.*n*	Integer32	dbStatMapPages	MapPages
1.3.6.1.4.1.897.2.2.1.44.*n*	Integer32	dbStatMaxIO	MaxIO
1.3.6.1.4.1.897.2.2.1.45.*n*	Counter64	dbStatMaxRead	MaxRead
1.3.6.1.4.1.897.2.2.1.46.*n*	Integer32	dbStatMaxWrite	MaxWrite
1.3.6.1.4.1.897.2.2.1.47.*n*	Integer32	dbStatPageRelocations	PageRelocations

OID	型	名前	プロパティ
1.3.6.1.4.1.897.2.2.1.48.*n*	Integer32	dbStatProcedurePages	ProcedurePages
1.3.6.1.4.1.897.2.2.1.49.*n*	Integer32	dbStatQueryCachePages	QueryCachePages
1.3.6.1.4.1.897.2.2.1.50.*n*	Integer32	dbStatQueryLowMemoryStrategy	QueryLowMemoryStrategy
1.3.6.1.4.1.897.2.2.1.51.*n*	Counter64	dbStatQueryRowsMaterialized	QueryRowsMaterialized
1.3.6.1.4.1.897.2.2.1.52.*n*	Integer32	dbStatRecoveryUrgency	RecoveryUrgency
1.3.6.1.4.1.897.2.2.1.53.*n*	Integer32	dbStatLogFreeCommit	LogFreeCommit
1.3.6.1.4.1.897.2.2.1.54.*n*	Integer32	dbStatLogWrite	LogWrite
1.3.6.1.4.1.897.2.2.1.55.*n*	Integer32	dbStatRelocatableHeapPages	RelocatableHeapPages
1.3.6.1.4.1.897.2.2.1.56.*n*	Integer32	dbStatRollbackLogPages	RollbackLogPages
1.3.6.1.4.1.897.2.2.1.57.*n*	Integer32	dbStatTempTablePages	TempTablePages
1.3.6.1.4.1.897.2.2.1.58.*n*	Integer32	dbStatTriggerPages	TriggerPages
1.3.6.1.4.1.897.2.2.1.59.*n*	Integer32	dbStatViewPages	ViewPages
1.3.6.1.4.1.897.2.2.1.60.*n*	Integer32	dbStatVersionStorePages	VersionStorePages
1.3.6.1.4.1.897.2.2.1.61.*n*	Integer32	dbStatSnapshotCount	SnapshotCount
1.3.6.1.4.1.897.2.2.1.62.*n*	Integer32	dbStatLockCount	LockCount

SQL Anywhere MIB データベース・プロパティ

次の表は、SQL Anywhere SNMP Extension Agent を使用して検索できるデータベース・プロパティの OID と名前を示します。

書き込み可能なプロパティには、アスタリスク記号 (*) が付いています。値 *n* は、*sasnmp.ini* ファイル内のデータベース番号です。

データベース・プロパティの詳細については、「データベース・レベルのプロパティ」 550 ページを参照してください。

OID	型	名前	プロパティ
1.3.6.1.4.1.897.2.2.2.1.*n*	String	dbPropAlias	Alias
1.3.6.1.4.1.897.2.2.2.2.*n*	String	dbPropAuditingTypes	AuditingTypes
1.3.6.1.4.1.897.2.2.2.3.*n*	String	dbPropBlankPadding	BlankPadding
1.3.6.1.4.1.897.2.2.2.4.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.5.*n*	String	dbPropCapabilities	Capabilities
1.3.6.1.4.1.897.2.2.2.6.*n*	String	dbPropCaseSensitive	CaseSensitive
1.3.6.1.4.1.897.2.2.2.7.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.8.*n*	String	dbPropCharSet	CharSet
1.3.6.1.4.1.897.2.2.2.9.*n*	String	dbPropChecksum	Checksum
1.3.6.1.4.1.897.2.2.2.10.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.11.*n*	String	dbPropCollation	Collation
1.3.6.1.4.1.897.2.2.2.12.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.13.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.14.*n*	String	dbPropCurrentRedoPos	CurrentRedoPos
1.3.6.1.4.1.897.2.2.2.15.*n*	String	dbPropDBFileFragments	DBFileFragments
1.3.6.1.4.1.897.2.2.2.16.*n*	String	dbPropDriveType	DriveType
1.3.6.1.4.1.897.2.2.2.17.*n*	String	dbPropEncryption	Encryption
1.3.6.1.4.1.897.2.2.2.18.*n*	String	dbPropFile	File
1.3.6.1.4.1.897.2.2.2.19.*n*	String	dbPropFileSize	FileSize
1.3.6.1.4.1.897.2.2.2.20.*n*	String	dbPropInterna	Internal
1.3.6.1.4.1.897.2.2.2.21.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.22.*n*	String	dbPropFreePages	FreePages
1.3.6.1.4.1.897.2.2.2.23.*n*	String	dbPropGlobalDBID	GlobalDBID
1.3.6.1.4.1.897.2.2.2.24.*n*	String	dbPropInternal	Internal

OID	型	名前	プロパティ
1.3.6.1.4.1.897.2.2.2.25.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.26.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.27.*n*	String	dbPropIQStore	IQStore
1.3.6.1.4.1.897.2.2.2.28.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.29.*n*	String	dbPropLanguage	Language
1.3.6.1.4.1.897.2.2.2.30.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.31.*n*	String	dbPropLogFileFragments	LogFileFragments
1.3.6.1.4.1.897.2.2.2.32.*n*	String	dbPropLogMirrorName	LogMirrorName
1.3.6.1.4.1.897.2.2.2.33.*n*	String	dbPropLogName	LogName
1.3.6.1.4.1.897.2.2.2.34.*n*	String	dbPropLTMGeneration	LTMGeneration
1.3.6.1.4.1.897.2.2.2.35.*n*	String	dbPropLTMTrunc	LTMTrunc
1.3.6.1.4.1.897.2.2.2.36.*n*	String	dbPropMultiByteCharSet	MultiByteCharSet
1.3.6.1.4.1.897.2.2.2.37.*n*	String	dbPropName	Name
1.3.6.1.4.1.897.2.2.2.38.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.39.*n*	String	dbPropPageSize	PageSize
1.3.6.1.4.1.897.2.2.2.40.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.41.*n*	String	dbPropProcedureProfiling*	ProcedureProfiling
1.3.6.1.4.1.897.2.2.2.42.*n*	String	dbPropReadOnly	ReadOnly
1.3.6.1.4.1.897.2.2.2.43.*n*	String	dbPropRemoteTrunc	RemoteTrunc
1.3.6.1.4.1.897.2.2.2.44.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.45.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.46.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.47.*n*	String	dbPropSyncTrunc	SyncTrunc
1.3.6.1.4.1.897.2.2.2.48.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.49.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.50.*n*	String	dbPropTempFileName	TempFileName
1.3.6.1.4.1.897.2.2.2.51.*n*	String	dbPropInternal	Internal

OID	型	名前	プロパティ
1.3.6.1.4.1.897.2.2.2.52.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.53.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.54.*n*	String	dbPropNextScheduleTime	NextScheduleTime
1.3.6.1.4.1.897.2.2.2.55.*n*	String	dbPropIdentitySignature	IdentitySignature
1.3.6.1.4.1.897.2.2.2.56.*n*	String	dbPropInternal	Internal
1.3.6.1.4.1.897.2.2.2.57.*n*	String	dbPropSnapshotIsolationState	SnapshotIsolationState
1.3.6.1.4.1.897.2.2.2.58.*n*	String	dbPropConnsDisabled	ConnsDisabled
1.3.6.1.4.1.897.2.2.2.59.*n*	String	dbPropPartnerState	PartnerState
1.3.6.1.4.1.897.2.2.2.60.*n*	String	dbPropArbiterState	ArbiterState
1.3.6.1.4.1.897.2.2.2.61.*n*	String	dbPropMirrorState	MirrorState
1.3.6.1.4.1.897.2.2.2.62.*n*	String	dbPropAlternateServerName	AlternateServerName
1.3.6.1.4.1.897.2.2.2.63.*n*	String	dbPropEncryptionScope	EncryptionScope
1.3.6.1.4.1.897.2.2.2.64.*n*	String	dbPropNcharCharSet	NcharCharSet
1.3.6.1.4.1.897.2.2.2.65.*n*	String	dbPropNcharCollation	NcharCollation
1.3.6.1.4.1.897.2.2.2.66.*n*	String	dbPropAccentSensitive	AccentSensitive
1.3.6.1.4.1.897.2.2.2.67.*n*	String	dbPropSendingTracingTo	SendingTracingTo
1.3.6.1.4.1.897.2.2.2.68.*n*	String	dbPropReceivingTracingFrom	ReceivingTracingFrom
1.3.6.1.4.1.897.2.2.2.69.*n*	String	dbPropIOParallelism	IOParallelism
1.3.6.1.4.1.897.2.2.2.70.*n*	String	dbPropJavaVM	JavaVM
1.3.6.1.4.1.897.2.2.2.71.*n*	String	dbPropDatabaseCleaner	DatabaseCleaner
1.3.6.1.4.1.897.2.2.2.72.*n*	String	dbPropHasCollationTailoring	HasCollationTailoring
1.3.6.1.4.1.897.2.2.2.73.*n*	String	dbPropCatalogCollation	CatalogCollation

SQL Anywhere MIB データベース・オプション

次の表は、SQL Anywhere SNMP Extension Agent を使用して検索できるデータベース・オプションの OID と名前を示します。

書き込み可能なオプションには、アスタリスク記号 (*) が付いています。値 *n* は、*sasnmp.ini* ファイル内のデータベース番号です。

データベース・オプションの詳細については、「アルファベット順のオプション・リスト」 421 ページを参照してください。

OID	型	名前	オプション
1.3.6.1.4.1.897.2.2.3.1.*n*	String	dbOptAllowNullsByDefault*	allow_nulls_by_default
1.3.6.1.4.1.897.2.2.3.2.*n*	String	dbOptAnsinull*	ansinull
1.3.6.1.4.1.897.2.2.3.3.*n*	String	dbOptAnsiBlanks*	ansi_blanks
1.3.6.1.4.1.897.2.2.3.4.*n*	String	dbOptAnsiCloseCursorsOnRollback*	ansi_close_cursors_on_rollback
1.3.6.1.4.1.897.2.2.3.5.*n*	String	dbOptAnsiIntegerOverflow*	ansi_integer_overflow
1.3.6.1.4.1.897.2.2.3.6.*n*	String	dbOptAnsiPermissions*	ansi_permissions
1.3.6.1.4.1.897.2.2.3.7.*n*	String	dbOptAnsiUpdateConstraints*	ansi_update_constraints
1.3.6.1.4.1.897.2.2.3.8.*n*	String	dbOptAuditing*	auditing
1.3.6.1.4.1.897.2.2.3.9.*n*	String	dbOptAuditingOptions*	auditing_options
1.3.6.1.4.1.897.2.2.3.10.*n*	String	dbOptAutomaticTimestamp*	automatic_timestamp
1.3.6.1.4.1.897.2.2.3.11.*n*	String	dbOptBackgroundPriority*	background_priority
1.3.6.1.4.1.897.2.2.3.12.*n*	String	dbOptBlocking*	blocking
1.3.6.1.4.1.897.2.2.3.13.*n*	Integer32	dbOptBlockingTimeout*	blocking_timeout
1.3.6.1.4.1.897.2.2.3.14.*n*	String	dbOptChained*	chained
1.3.6.1.4.1.897.2.2.3.15.*n*	Integer32	dbOptCheckpointTime*	checkpoint_time
1.3.6.1.4.1.897.2.2.3.16.*n*	Integer32	dbOptCisOption*	cis_option
1.3.6.1.4.1.897.2.2.3.17.*n*	Integer32	dbOptCisRowsetSize*	cis_rowset_size
1.3.6.1.4.1.897.2.2.3.18.*n*	String	dbOptCloseOnEndtrans*	close_on_endtrans
1.3.6.1.4.1.897.2.2.3.19.*n*	String	dbOptConnectionAuthentication*	connection_authentication

OID	型	名前	オプション
1.3.6.1.4.1.897.2.2.3.20.*n*	String	dbOptContinueAfterRaiseerror*	continue_after_raise_error
1.3.6.1.4.1.897.2.2.3.21.*n*	String	dbOptConversionError*	conversion_error
1.3.6.1.4.1.897.2.2.3.22.*n*	String	dbOptCooperativeCommits*	cooperative_commits
1.3.6.1.4.1.897.2.2.3.23.*n*	Integer32	dbOptCooperativeCommitTimeout*	cooperative_commit_timeout
1.3.6.1.4.1.897.2.2.3.24.*n*	String	dbOptDatabaseAuthentication*	database_authentication
1.3.6.1.4.1.897.2.2.3.25.*n*	String	dbOptDateFormat*	date_format
1.3.6.1.4.1.897.2.2.3.26.*n*	String	dbOptDateOrder*	date_order
1.3.6.1.4.1.897.2.2.3.27.*n*	String	dbOptDebugMessages*	debug_messages
1.3.6.1.4.1.897.2.2.3.28.*n*	String	dbOptDedicatedTask*	dedicated_task
1.3.6.1.4.1.897.2.2.3.29.*n*	Integer32	dbOptDefaultTimestampIncrement*	default_timestamp_increment
1.3.6.1.4.1.897.2.2.3.30.*n*	String	dbOptDelayedCommits*	delayed_commits
1.3.6.1.4.1.897.2.2.3.31.*n*	Integer32	dbOptDelayedCommitTimeout*	delayed_commit_timeout
1.3.6.1.4.1.897.2.2.3.32.*n*	String	dbOptDivideByZeroError*	divide_by_zero_error
1.3.6.1.4.1.897.2.2.3.33.*n*	String	dbOptEscapeCharacter*	escape_character
1.3.6.1.4.1.897.2.2.3.34.*n*	String	dbOptExcludeOperators*	exclude_operators
1.3.6.1.4.1.897.2.2.3.35.*n*	String	dbOptExtendedJoinSyntax*	extended_join_syntax
1.3.6.1.4.1.897.2.2.3.36.*n*	String	dbOptFireTriggers*	fire_triggers
1.3.6.1.4.1.897.2.2.3.37.*n*	Integer32	dbOptFirstDayOfWeek*	first_day_of_week
1.3.6.1.4.1.897.2.2.3.38.*n*	String	dbOptFloatAsDouble*	float_as_double
1.3.6.1.4.1.897.2.2.3.39.*n*	String	dbOptForceViewCreation*	force_view_creation
1.3.6.1.4.1.897.2.2.3.40.*n*	String	dbOptForXmlNullTreatment*	for_xml_null_treatment
1.3.6.1.4.1.897.2.2.3.41.*n*	Integer32	dbOptGlobalDatabaseId*	global_database_id

OID	型	名前	オプション
1.3.6.1.4.1.897.2.2.3.42.*n*	Integer32	dbOptIsolationLevel*	isolation_level
1.3.6.1.4.1.897.2.2.3.43.*n*	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.44.*n*	String	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.45.*n*	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.46.*n*	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.47.*n*	String	dbOptLockRejectedRows*	lock_rejected_rows
1.3.6.1.4.1.897.2.2.3.48.*n*	String	dbOptLoginMode*	login_mode
1.3.6.1.4.1.897.2.2.3.49.*n*	String	dbOptLoginProcedure*	login_procedure
1.3.6.1.4.1.897.2.2.3.50.*n*	String	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.51.*n*	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.52.*n*	Integer32	dbOptMaxCursorCount*	max_cursor_count
1.3.6.1.4.1.897.2.2.3.53.*n*	Integer32	dbOptMaxHashSize*	max_hash_size
1.3.6.1.4.1.897.2.2.3.54.*n*	Integer32	dbOptMaxPlansCached*	max_plans_cached
1.3.6.1.4.1.897.2.2.3.55.*n*	Integer32	dbOptMaxRecursiveIterations*	max_recursive_iterations
1.3.6.1.4.1.897.2.2.3.56.*n*	Integer32	dbOptMaxStatementCount*	max_statement_count
1.3.6.1.4.1.897.2.2.3.57.*n*	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.58.*n*	Integer32	dbOptMinPasswordLength*	min_password_length
1.3.6.1.4.1.897.2.2.3.59.*n*	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.60.*n*	Integer32	dbOptNearestCentury*	nearest_century
1.3.6.1.4.1.897.2.2.3.61.*n*	String	dbOptNonKeywords*	non_keywords

OID	型	名前	オプション
1.3.6.1.4.1.897.2.2.3.62.*n*	String	dbOptODBCDistinguishCharAndVarchar*	odbc_distinguish_char_and_varchar
1.3.6.1.4.1.897.2.2.3.63.*n*	String	dbOptOnCharsetConversionFailure*	on_charset_conversion_failure
1.3.6.1.4.1.897.2.2.3.64.*n*	String	dbOptOnTsqlError*	on_tsql_error
1.3.6.1.4.1.897.2.2.3.65.*n*	String	dbOptOptimisticWaitForCommit*	optimistic_wait_for_commit
1.3.6.1.4.1.897.2.2.3.66.*n*	String	dbOptOptimizationGoal*	optimization_goal
1.3.6.1.4.1.897.2.2.3.67.*n*	Integer32	dbOptOptimizationLevel*	optimization_level
1.3.6.1.4.1.897.2.2.3.68.*n*	String	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.69.*n*	String	dbOptOptimizationWorkload*	optimization_workload
1.3.6.1.4.1.897.2.2.3.70.*n*	Integer32	dbOptPinnedCursorPercentOfCache*	pinned_cursor_percent_of_cache
1.3.6.1.4.1.897.2.2.3.71.*n*	Integer32	dbOptPrecision*	precision
1.3.6.1.4.1.897.2.2.3.72.*n*	String	dbOptPrefetch*	prefetch
1.3.6.1.4.1.897.2.2.3.73.*n*	String	dbOptPreserveSourceFormat*	preserve_source_format
1.3.6.1.4.1.897.2.2.3.74.*n*	String	dbOptPreventArticlePkeyUpdate*	prevent_article_pkey_update
1.3.6.1.4.1.897.2.2.3.75.*n*	String	dbOptQueryPlanOnOpen*	query_plan_on_open
1.3.6.1.4.1.897.2.2.3.76.*n*	String	dbOptQuotedIdentifier*	quoted_identifier
1.3.6.1.4.1.897.2.2.3.77.*n*	String	dbOptReadPastDeleted*	read_past_deleted
1.3.6.1.4.1.897.2.2.3.78.*n*	Integer32	dbOptRecoveryTime*	recovery_time
1.3.6.1.4.1.897.2.2.3.79.*n*	String	dbOptReplicateAll*	replicate_all
1.3.6.1.4.1.897.2.2.3.80.*n*	String	dbOptReturnDateTimeAsString*	return_date_time_as_string
1.3.6.1.4.1.897.2.2.3.81.*n*	String	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.82.*n*	String	dbOptRITriggerTime*	ri_trigger_time

OID	型	名前	オプション
1.3.6.1.4.1.897.2.2.3.83.*n*	String	dbOptRowCounts*	row_counts
1.3.6.1.4.1.897.2.2.3.84.*n*	Integer32	dbOptScale*	scale
1.3.6.1.4.1.897.2.2.3.85.*n*	String	dbOptSortCollation*	sort_collation
1.3.6.1.4.1.897.2.2.3.86.*n*	String	dbOptSQLFlaggerErrorLevel*	sql_flagger_error_level
1.3.6.1.4.1.897.2.2.3.87.*n*	String	dbOptSQLFlaggerWarningLevel*	sql_flagger_warning_level
1.3.6.1.4.1.897.2.2.3.88.*n*	String	dbOptStringRtruncation*	string_rtruncation
1.3.6.1.4.1.897.2.2.3.89.*n*	String	dbOptSubsumeRowLocks*	subsume_row_locks
1.3.6.1.4.1.897.2.2.3.90.*n*	String	dbOptSuppressTDSDebugging*	suppress_tds_debugging
1.3.6.1.4.1.897.2.2.3.91.*n*	String	dbOptTDSEmptyStringIsNull*	tds_empty_string_is_null
1.3.6.1.4.1.897.2.2.3.92.*n*	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.93.*n*	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.94.*n*	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.95.*n*	String	dbOptTimestampFormat*	timestamp_format
1.3.6.1.4.1.897.2.2.3.96.*n*	String	dbOptTimeFormat*	time_format
1.3.6.1.4.1.897.2.2.3.97.*n*	Integer32	dbOptTimeZoneAdjustment*	time_zone_adjustment
1.3.6.1.4.1.897.2.2.3.98.*n*	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.99.*n*	String	dbOptTruncateTimestampValues*	truncate_timestamp_values
1.3.6.1.4.1.897.2.2.3.100.*n*	String	dbOptTruncateWithAutocommit*	truncate_with_auto_commit
1.3.6.1.4.1.897.2.2.3.101.*n*	String	dbOptTsqlHexConstant*	tsql_hex_constant
1.3.6.1.4.1.897.2.2.3.102.*n*	String	dbOptTsqlVariables*	tsql_variables
1.3.6.1.4.1.897.2.2.3.103.*n*	String	dbOptUpdateStatistics*	update_statistics

OID	型	名前	オプション
1.3.6.1.4.1.897.2.2.3.104.*n*	String	dbOptUpgradeDatabaseCapability*	upgrade_database_capability
1.3.6.1.4.1.897.2.2.3.105.*n*	String	dbOptUserEstimates*	user_estimates
1.3.6.1.4.1.897.2.2.3.106.*n*	String	dbOptWaitForCommit*	wait_for_commit
1.3.6.1.4.1.897.2.2.3.107.*n*	String	dbOptTempSpaceLimitCheck*	temp_space_limit_check
1.3.6.1.4.1.897.2.2.3.108.*n*	Integer32	dbOptRemoteIdleTimeout*	remote_idle_timeout
1.3.6.1.4.1.897.2.2.3.109.*n*	Integer32	dbOptAnsiSubstring	ansi_substring
1.3.6.1.4.1.897.2.2.3.110.*n*	String	dbOptODBCDescribeBinaryAsVarbinary*	odbc_describe_binary_as_varbinary
1.3.6.1.4.1.897.2.2.3.111.*n*	String	dbOptRollbackOnDeadlock	rollback_on_deadlock
1.3.6.1.4.1.897.2.2.3.112.*n*	String	dbOptIntegratedServerName*	integrated_server_name
1.3.6.1.4.1.897.2.2.3.113.*n*	String	dbOptLogDeadlocks*	log_deadlocks
1.3.6.1.4.1.897.2.2.3.114.*n*	Integer32	dbOptInternal	Internal
1.3.6.1.4.1.897.2.2.3.115.*n*	String	dbOptWebserviceNamespaceHost*	webservice_namespace_host
1.3.6.1.4.1.897.2.2.3.116.*n*	Integer32	dbOptMaxQueryRequests*	max_query_tasks
1.3.6.1.4.1.897.2.2.3.117.*n*	Integer32	dbOptRequestTimeout*	request_timeout
1.3.6.1.4.1.897.2.2.3.118.*n*	String	dbOptSynchronizeMirrorOnCommit*	synchronize_mirror_on_commit
1.3.6.1.4.1.897.2.2.3.119.*n*	Integer32	dbOptHttpSessionTimeout*	http_session_timeout
1.3.6.1.4.1.897.2.2.3.120.*n*	String	dbOptUuidHasHyphens*	uuid_has_hyphens
1.3.6.1.4.1.897.2.2.3.121.*n*	String	dbOptAllowSnapshotIsolation*	allow_snapshot_isolation
1.3.6.1.4.1.897.2.2.3.122.*n*	String	dbOptVerifyPasswordFunction*	verify_password_function
1.3.6.1.4.1.897.2.2.3.123.*n*	String	dbOptDefaultDbspace*	default_dbspace

OID	型	名前	オプション
1.3.6.1.4.1.897.2.2.3.124.*n*	String	dbOptCollectStatisticsOnDmlUpdates*	collect_statistics_on_dml_updates
1.3.6.1.4.1.897.2.2.3.125.*n*	String	dbOptJavaMainUserid*	java_main_userid
1.3.6.1.4.1.897.2.2.3.126.*n*	String	dbOptJavaLocation*	java_location
1.3.6.1.4.1.897.2.2.3.127.*n*	String	dbOptOemString*	oem_string
1.3.6.1.4.1.897.2.2.3.128.*n*	Integer32	dbOptMaxTempSpace*	max_temp_space
1.3.6.1.4.1.897.2.2.3.129.*n*	String	dbOptSecureFeatureKey*	secure_feature_key
1.3.6.1.4.1.897.2.2.3.130.*n*	String	dbOptMaterializedViewOptimization*	materialized_view_optimization
1.3.6.1.4.1.897.2.2.3.131.*n*	Integer32	dbOptUpdatableStatementIsolation*	updatable_statement_isolation
1.3.6.1.4.1.897.2.2.3.132.*n*	String	dbOptTsqlOuterJoins*	tsql_outer_joins
1.3.6.1.4.1.897.2.2.3.133.*n*	String	dbOptPostLoginProcedure*	post_login_procedure
1.3.6.1.4.1.897.2.2.3.134.*n*	String	dbOptConnAuditing*	conn_auditing
1.3.6.1.4.1.897.2.2.3.135.*n*	String	dbOptPercentAsComment*	percent_as_comment
1.3.6.1.4.1.897.2.2.3.136.*n*	String	dbOptJavaVmOptions*	java_vm_options
1.3.6.1.4.1.897.2.2.3.137.*n*	String	dbOptEncryptAesRandomIv*	encrypt_aes_random_iv
1.3.6.1.4.1.897.2.2.3.138.*n*	Integer32	dbOptMaxClientStatementsCached*	max_client_statements_cached

第 19 章

RDBMS MIB リファレンス

目次

RDBMS MIB

以下の項では、SQL Anywhere SNMP Extension Agent を使用して検索できる値の OID を示します。デフォルトで、RDBMS MIB は *C:¥Program Files¥SQL Anywhere 10¥snmp¥RDBMS-MIB.mib* にあります。

rdbmsDbTable

このテーブルは、システムにインストールされているデータベースについての情報を示します。

値 *db* は、*sasnmp.ini* ファイル内のデータベース番号です。

OID	型	名前	戻り値
1.3.6.1.2.1.39.1.1.1.1.*db*	Integer	rdbmsDbIndex	*db*
1.3.6.1.2.1.39.1.1.1.2.*db*	OID	rdbmsDbPrivateMibOID	1.3.6.1.4.1.897.2
1.3.6.1.2.1.39.1.1.1.3.*db*	String	rdbmsDbVendorName	PROPERTY('CompanyName')
1.3.6.1.2.1.39.1.1.1.4.*db*	String	rdbmsDbName	DB_PROPERTY('Name')
1.3.6.1.2.1.39.1.1.1.5.*db*	String	rdbmsDbContact	PROPERTY('LicensedUser')

rdbmsDbInfoTable

このテーブルは、システム上のデータベースについての追加情報を示します。

値 *db* は、*sasnmp.ini* ファイル内のデータベース番号です。

OID	型	名前	戻り値
1.3.6.1.2.1.39.1.2.1.1.*db*	String	rdbmsDbInfoProductName	PROPERTY('ProductName')
1.3.6.1.2.1.39.1.2.1.2.*db*	String	rdbmsDbInfoVersion	PROPERTY('ProductVersion')

OID	型	名前	戻り値
1.3.6.1.2.1.39.1.2.1.3.*db*	Integer	rdbmsDbInfoSizeUnits	dbInfoSizeAllocated と dbInfoSizeUsed に基づいて計算される ◆ 1= バイト ◆ 2=KB ◆ 3=MB ◆ 4=GB ◆ 5=TB (各単位はその 1 つ前の単位の 1024 倍)
1.3.6.1.2.1.39.1.2.1.4.*db*	Integer	rdbmsDbInfoSizeAllocated	DB_PROPERTY('PageSize') * DB_PROPERTY('FileSize')
1.3.6.1.2.1.39.1.2.1.5.*db*	Integer	rdbmsDbInfoSizeUsed	DB_PROPERTY('PageSize') * (DB_PROPERTY('FileSize') - DB_PROPERTY('FreePages'))
1.3.6.1.2.1.39.1.2.1.6.*db*	String	rdbmsDbInfoLastBackup	NULL[1]

[1] この OID は SQL Anywhere SNMP Extension Agent ではサポートされていません。

rdbmsDbParamTable

このテーブルは、システム上のデータベースの設定パラメータを示します。

値 *db* は *sasnmp.ini* ファイル内のデータベース番号です。また、*n* は sa.2.3 サブツリー内のオプションのインデックスです。

OID	型	名前	戻り値
1.3.6.1.2.1.39.1.3.1.1.*db*	String	rdbmsDbParamName	オプション名
1.3.6.1.2.1.39.1.3.1.2.*db*	Integer	rdbmsDbParamSubIndex	*n*
1.3.6.1.2.1.39.1.3.1.3.*db*	OID	rdbmsDbParamID	このオプションに対応する SQL Anywhere MIB 内の OID
1.3.6.1.2.1.39.1.3.1.4.*db*	String	rdbmsDbParamCurrValue	オプション値
1.3.6.1.2.1.39.1.3.1.5.*db*	String	rdbmsDbParamComment	NULL[1]

[1] この OID は SQL Anywhere SNMP Extension Agent ではサポートされていません。

rdbmsDbLimitedResourceTable

このテーブルは、各 DB 領域の空き領域についての情報を示します。表中の *n* は、以下に示すとおり、いずれかの DB 領域を表す値になります。

- 1 ー 13：通常の DB 領域 (データベース内では番号 0 ー 12)
- 14：トランザクション・ログ・ファイル
- 15：トランザクション・ログ・ミラー・ファイル
- 16：テンポラリ・ファイル

値 *db* は、*sasnmp.ini* ファイル内のデータベース番号です。

OID	型	名前	戻り値
1.3.6.1.2.1.39.1.4.1.1.*n.db*	String	rdbmsDbLimitedResourceName	DB 領域の名前、トランザクション・ログ、トランザクション・ログ・ミラー、またはテンポラリ・ファイル
1.3.6.1.2.1.39.1.4.1.2.*n.db*	OID	rdbmsDbLimitedResourceID	1.3.6.1.4.1.897.2
1.3.6.1.2.1.39.1.4.1.3.*n.db*	Integer	rdbmsDbLimitedResourceLimit	ディスク上の使用可能な空き領域 + 現在のファイル・サイズ
1.3.6.1.2.1.39.1.4.1.4.*n.db*	Integer	rdbmsDbLimitedResourceCurrent	現在のファイル・サイズ
1.3.6.1.2.1.39.1.4.1.5.*n.db*	Integer	rdbmsDbLimitedResourceHighwater	現在のサイズ
1.3.6.1.2.1.39.1.4.1.6.*n.db*	Integer	rdbmsDbLimitedResourceFailure	0[1]
1.3.6.1.2.1.39.1.4.1.7.*n.db*	String	rdbmsDbLimitedResourceDescription	バイト、KB、MB、GB、TB のいずれか

[1] この OID は SQL Anywhere SNMP Extension Agent ではサポートされていません。

rdbmsSrvTable

このテーブルは、システム上の動作中またはインストール済みのデータベース・サーバを示します。

値 *db* は、*sasnmp.ini* ファイル内のデータベース番号です。

OID	型	名前	戻り値
1.3.6.1.2.1.39.1.5.1.1.*db*	OID	rdbmsSrvPrivateMibOID	1.3.6.1.4.1.897.2
1.3.6.1.2.1.39.1.5.1.2.*db*	String	rdbmsSrvVendorName	PROPERTY('CompanyName')
1.3.6.1.2.1.39.1.5.1.3.*db*	String	rdbmsSrvProductName	PROPERTY('ProductName')
1.3.6.1.2.1.39.1.5.1.4.*db*	String	rdbmsSrvContact	PROPERTY ('LicensedCompany')

rdbmsSrvInfoTable

このテーブルは、システム内のデータベース・サーバについての追加情報を示します。

値 *db* は、*sasnmp.ini* ファイル内のデータベース番号です。

OID	型	名前	戻り値
1.3.6.1.2.1.39.1.6.1.1.*db*	Integer	rdbmsSrvInfoStartupTime	PROPERTY ('StartTime')
1.3.6.1.2.1.39.1.6.1.2.*db*	Integer	rdbmsSrvInfoFinishedTransactions	0[1]
1.3.6.1.2.1.39.1.6.1.3.*db*	Integer	rdbmsSrvInfoDiskReads	PROPERTY ('DiskReadEng')
1.3.6.1.2.1.39.1.6.1.4.*db*	Integer	rdbmsSrvInfoLogicalReads	0[1]
1.3.6.1.2.1.39.1.6.1.5.*db*	Integer	rdbmsSrvInfoDiskWrites	PROPERTY ('DiskWriteEng')
1.3.6.1.2.1.39.1.6.1.6.*db*	Integer	rdbmsSrvInfoLogicalWrites	0[1]
1.3.6.1.2.1.39.1.6.1.7.*db*	Integer	rdbmsSrvInfoPageReads	0[1]
1.3.6.1.2.1.39.1.6.1.8.*db*	Integer	rdbmsSrvInfoPageDiskOutOfWrites	0[1]
1.3.6.1.2.1.39.1.6.1.9.*db*	Integer	rdbmsSrvInfoSpaces	0[1]
1.3.6.1.2.1.39.1.6.1.10.*db*	Integer	rdbmsSrvInfoHandledRequests	PROPERTY('Req')
1.3.6.1.2.1.39.1.6.1.11.*db*	Integer	rdbmsSrvInfoRequestRecvs	PROPERTY ('PacketsReceivedUncomp')
1.3.6.1.2.1.39.1.6.1.12.*db*	Integer	rdbmsSrvInfoRequestSends	PROPERTY ('PacketsSentUncomp')
1.3.6.1.2.1.39.1.6.1.13.*db*	Integer	rdbmsSrvInfoHighwaterInboundAssociations	0[1]

OID	型	名前	戻り値
1.3.6.1.2.1.39.1.6.1.14.*db*	Integer	rdbmsSrvInfoMaxInboundAssociations	0[1]

[1] この OID は SQL Anywhere SNMP Extension Agent ではサポートされていません。

rdbmsSrvParamTable

このテーブルは、SQL Anywhere SNMP Extension Agent で SQL Anywhere MIB を使用して設定できるサーバ・オプションを示します。*n* はインデックスです。このインデックスの詳細は次のとおりです。

n	サーバ・オプション
1	ConnsDisabled
2	LivenessTimeout (デフォルト)
3	QuittingTime
4	RememberLastStatement
5	RequestLogFile
6	RequestLogging

値 *db* は、*sasnmp.ini* ファイル内のデータベース番号です。

OID	型	名前	戻り値
1.3.6.1.2.1.39.1.7.1.1.*n.db*	String	rdbmsDbSrvParamName	オプション *n* の名前
1.3.6.1.2.1.39.1.7.1.2.*n.db*	Integer	rdbmsDbSrvParamSubIndex	*n*
1.3.6.1.2.1.39.1.7.1.3.*n.db*	OID	rdbmsDbSrvParamID	1.3.6.1.4.1.897.2
1.3.6.1.2.1.39.1.7.1.4.*n.db*	String	rdbmsDbSrvParamCurrValue	オプション *n* の現在の値
1.3.6.1.2.1.39.1.7.1.5.*n.db*	String	rdbmsDbSrvParamComment	オプション *n* の完全名

rdbmsSrvLimitedResourceTable

このテーブルには、サーバ設定パラメータについての情報が格納されています。

値 *db* は、*sasnmp.ini* ファイル内のデータベース番号です。また、*n* はリソースのインデックスです。このインデックスの詳細は次のとおりです。

n	名前	リソース	リソースの制限
1	Connections	PROPERTY ('UniqueClientAddresses')	PROPERTY('LicenseCount')
2	Processors	PROPERTY ('NumLogicalProcessorsUsed')	PROPERTY ('NumLogicalProcessorsUsed')

OID	型	名前	戻り値
1.3.6.1.2.1.39.1.8.1.1.*db*	String	rdbmsSrvLimitedResource Name	リソース *n* の名前
1.3.6.1.2.1.39.1.8.1.2.*db*	OID	rdbmsSrvLimitedResource ID	このオプションに対応する SQL Anywhere MIB 内の OID
1.3.6.1.2.1.39.1.8.1.3.*db*	Integer	rdbmsSrvLimitedResource Limit	リソース *n* の上限
1.3.6.1.2.1.39.1.8.1.4.*db*	Integer	rdbmsSrvLimitedResource Current	リソース *n* の現在の値
1.3.6.1.2.1.39.1.8.1.5.*db*	Integer	rdbmsSrvLimitedResource Highwater	リソース *n* の現在の値
1.3.6.1.2.1.39.1.8.1.6.*db*	Integer	rdbmsSrvLimitedResource Failures	0[1]
1.3.6.1.2.1.39.1.8.1.7.*db*	String	rdbmsSrvLimitedResource Description	リソース *n* の名前

[1] この OID は SQL Anywhere SNMP Extension Agent ではサポートされていません。

パート V. データベースの保守

パート V では、データベース・ファイルのバックアップ方法と、イベントやスケジュールを使用してデータベース管理を自動化する方法について説明します。

第 20 章

バックアップとデータ・リカバリ

目次

バックアップとリカバリの概要

「**バックアップ**」とはデータベースにある情報のコピーであり、元のデータベースとは物理的に別の場所に格納されます。ディスク・ドライブの破損などの理由からデータベースが使用不可能になった場合、バックアップしたデータベースを「**リストア**」できます。発生した障害の性質によって異なりますが、使用不可能になった時点までにデータベースでコミットされていたすべての変更のバックアップをリストアできる場合もあります。

オペレーティング・システムまたはデータベース・サーバがクラッシュした場合、またはデータベース・サーバが正常に停止されなかった場合に、リカバリが発生します。データベース・サーバは起動時に、直前のセッションの最後にデータベースが正しく停止されたかどうかをチェックします。正しく停止されていない場合、サーバは自動リカバリ処理を実行して情報をリストアします。このメカニズムでは、最後にコミットされたトランザクションまでのすべての変更がリカバリされます。

質問と回答

質問…	**参照先…**
バックアップとは？	「バックアップとリカバリの概要」 812 ページ
リカバリとは？	「バックアップとリカバリの概要」 812 ページ
トランザクション・ログとは？	「トランザクション・ログ」 816 ページ
メディア障害とシステム障害とは？	「障害からのデータの保護」 814 ページ
どの種類の障害からデータを保護するためにバックアップするのか？	「障害からのデータの保護」 814 ページ
バックアップに使用できるツールは？	「バックアップの作成方法」 814 ページ
使用可能なバックアップの種類は？	「バックアップの種類」 819 ページ
適切なバックアップの種類は？	「バックアップ・プロシージャの設計」 819 ページ
データベース・ファイルまたはトランザクション・ログが破損した場合に失われるデータは？	「メディア障害からのデータの保護」 818 ページ
バックアップはどのように実行されるか？	「バックアップの概要」 816 ページ
バックアップの実行頻度は？	「バックアップのスケジュール」 820 ページ
自動バックアップをスケジュールできるか？	「バックアップのスケジュール」 820 ページ

質問…	参照先…
データベースがレプリケーションに関連する場合、バックアップ方式へどのように影響するか？	「レプリケーションに関連するデータベースのバックアップ・スキーマ」 824 ページ 「レプリケーション・インストールにおけるリモート・データベースのバックアップ方法」 826 ページ
テープにバックアップする方法は？	「データベースを直接テープにバックアップする」 855 ページ
バックアップ・スケジュールを計画する方法は？	「バックアップとリカバリのプランの設計」 827 ページ
バックアップを自動化できるか？	「スケジュールとイベントの使用によるタスクの自動化」 869 ページ
データベース・ファイルが破損していないことを確認する方法は？	「データベースの妥当性の確認」 828 ページ 「データベースの検証」 847 ページ
トランザクション・ログが破損していないことを確認する方法は？	「データベースの開始時にトランザクション・ログを検証する」 843 ページ 「トランザクション・ログの検証」 848 ページ
障害に対して最大限データ保護を配慮したデータベースの実行方法は？	「データ保護を目的としたデータベースの構成」 830 ページ
高可用性とマシンの冗長性を確実にする方法は？	「コンピュータ全体に及ぶ障害からの保護」 831 ページ 「ライブ・バックアップの作成」 856 ページ
バックアップを実行する方法は？	「フル・バックアップの実行」 845 ページ
障害が発生した場合に、バックアップしたデータをリストアする方法は？	「データベース・ファイルのメディア障害からリカバリする」 857 ページ 「ミラーされていないトランザクション・ログのメディア障害からリカバリする」 858 ページ 「トランザクション・ログ・ミラーのメディア障害からリカバリする」 859 ページ
データベースのメンテナンス・プランを作成する方法は？	「メンテナンス・プランの作成」 868 ページ

障害からのデータの保護

データベースが使用できなくなった場合、データベースの「**障害**」が発生しています。SQL Anywhere では、次の種類の障害に対する防護策を備えています。

メディア障害　データベース・ファイルまたはトランザクション・ログが使用できない状態を指します。これは、データベース・ファイルを格納しているファイル・システムまたはデバイスが使用できなくなった場合や、ファイルが破損した場合などに発生します。

次に例を示します。

- データベース・ファイルまたはトランザクション・ログ・ファイルが格納されているディスク・ドライブが使用できなくなった場合。
- データベース・ファイルまたはトランザクション・ログ・ファイルが破損した場合。これはハードウェアまたはソフトウェアに問題があるために発生します。

バックアップによって、メディア障害からデータを保護できます。

「バックアップの概要」 816 ページを参照してください。

システム障害　トランザクションの途中で、コンピュータまたはオペレーティング・システムが停止した場合に発生します。これは、正しい手順でコンピュータを終了または再起動しなかったり、他のアプリケーションによってオペレーティング・システムがクラッシュしたり、停電した場合に起こる可能性があります。

次に例を示します。

- トランザクションが完了していないときに、停電やオペレーティング・システムのクラッシュ、またはコンピュータの不適切な再起動が原因で、コンピュータまたはオペレーティング・システムが一時的に使用できなくなった場合

システム障害が発生した後、次にデータベースを起動するときに、データベース・サーバは自動的にリカバリします。システム・エラー以前にコミットされたトランザクションの結果は保存されます。システム障害の前にコミットされていなかったトランザクションによる変更はすべてキャンセルされます。

「バックアップとリカバリの内部」 834 ページを参照してください。

コミットされなかった変更を手動でリカバリすることもできます。「コミットされていない操作のリカバリ」 862 ページを参照してください。

バックアップの作成方法

バックアップ方法には複数の種類があります。この項では主な方法については説明しますが、それぞれのオプションについては説明しません。

次の方法でバックアップを作成できます。

- **Sybase Central**　Sybase Central の [データベース・バックアップ] ウィザードと [バックアップ・イメージ作成] ウィザードを使用して、バックアップを作成できます。これらのウィザー

ドを表示するには、データベースを選択して [ファイル] メニュー (またはポップアップ・メニュー) の [データベースのバックアップ] または [バックアップ・イメージの作成] をクリックします。「データベースを直接テープにバックアップする」 855 ページを参照してください。

Sybase Central の [メンテナンス・プラン作成] ウィザードを使用して、データベースを定期的に検証およびバックアップすることもできます。「メンテナンス・プランの作成」 868 ページを参照してください。

- **バックアップ・ユーティリティ** dbbackup ユーティリティは、クライアント側のバックアップを作成します。たとえば、コマンド・プロンプトで次のコマンドを実行すると、データベースとトランザクション・ログのバックアップ・コピーがクライアント・コンピュータ上のディレクトリ *c:¥backup* に作成されます。

  ```
  dbbackup -c "connection-string" c:¥backup
  ```

  -s オプションを指定すると、データベースとトランザクション・ログのバックアップ・コピーをサーバ・コンピュータ上のディレクトリ *c:¥backup* に作成できます。

  ```
  dbbackup -c "connection-string" -s c:¥backup
  ```

  「バックアップ・ユーティリティ (dbbackup)」 640 ページを参照してください。

- **SQL 文** SQL 文を使用して、データベース・サーバでサーバ側のバックアップ操作を実行できます。たとえば、次の文を発行すると、データベース・ファイルとトランザクション・ログのバックアップ・コピーがサーバ・コンピュータのディレクトリ *c:¥backup* に保存されます。

  ```
  BACKUP DATABASE
  DIRECTORY 'c:¥¥backup';
  ```

  「BACKUP 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  dbbackup -s オプションを使用して BACKUP DATABASE 文を発行することもできます。「バックアップ・ユーティリティ (dbbackup)」 640 ページを参照してください。

- **オフライン・バックアップ** 上の例はすべて稼働中のデータベースに対して実行するオンライン・バックアップです。データベースが稼働していないときは、データベース・ファイルをコピーしてオフライン・バックアップを作成できます。

  「バックアップの種類」 819 ページを参照してください。

注意

データベースのオンライン・バックアップを作成するには、BACKUP 権限または REMOTE DBA 権限が必要です。

バックアップの概要

バックアップする必要があるファイルを判断したり、バックアップしたデータベースのリストア方法を理解したりするには、データベースに加えた変更がどのようにディスクに保存されるかを理解する必要があります。

データベース・ファイル

データベースが正常に停止された場合、データベース・ファイルにはデータベースにある全データの現行の完全なコピーが保持されています。しかし、データベースの稼働中は、データベース・ファイルは通常現行のものまたは完全なものではありません。

データベース・ファイルが全データの完全な現行のコピーを保持していることが保証されるのは、チェックポイントの完了直後だけです。チェックポイントの後は、データベース・キャッシュの全内容がディスク上にあります。

データベース・サーバは、次の場合にデータベースに対するチェックポイントを実行します。

- データベースの停止操作の一環として
- 最後にチェックポイントが行われてからの時間が、データベース・オプションの checkpoint_time を超えたとき
- リカバリ予想時間がデータベース・オプションの recovery_time を超えたとき
- データベース・サーバのアイドル状態が、ダーティ・ページをすべて書き込めるほどに長く続いているとき
- 接続により CHECKPOINT 文が発行されたとき
- データベース・サーバがトランザクション・ログなしで稼働している場合に、トランザクションがコミットされたとき

チェックポイントとチェックポイントの間には、データベース・ファイルに加え、トランザクション・ログという別のファイルも必要です。これらのファイルを使用することで、コミットされたすべてのトランザクションの完全なコピーを確実に保持できます。

参照

- 「チェックポイントとチェックポイント・ログ」 835 ページ
- 「データベース・サーバがチェックポイントのタイミングを決定する方法」 842 ページ

トランザクション・ログ

「**トランザクション・ログ**」は、データベース・ファイルとは別のファイルです。トランザクション・ログは、データベースに対して行われたすべての変更を格納します。挿入、更新、削除、コミット、ロールバック、データベース・スキーマの変更が、すべて記録されます。トランザクション・ログは、「**転送ログ**」または「**再実行ログ**」とも呼ばれます。

トランザクション・ログは、バックアップとリカバリの重要なコンポーネントであり、SQL Remote、Replication Agent、またはデータベース・ミラーリングを使用したデータ・レプリケーションにも不可欠です。

デフォルトでは、すべてのデータベースがトランザクション・ログを使用します。トランザクション・ログの使用はオプションですが、特別の理由がないかぎり、トランザクション・ログの使用をおすすめします。データベースの実行時にトランザクション・ログを使用すると、障害からの保護をより確実に行えるばかりでなく、パフォーマンスの向上が見込めます。

データベース・ファイルとトランザクション・ログを、コンピュータの別々のディスクに保存することをおすすめします。DB 領域とトランザクション・ログが同じディスク上にある状態でディスク障害が発生すると、すべて失われる可能性があります。しかし、データベースとトランザクション・ログが別々のディスクに保存されている場合は、すべてまたはほとんどのデータをリカバリできます。これは、ディスク障害が発生した場合でも、フル・データベースかトランザクション・ログ (これをもとにデータベースをリカバリできる) のいずれかが存在するからです。

「データベース・ファイルで発生したメディア障害からの保護」 830 ページを参照してください。

警告
データベース・ファイルとトランザクション・ログ・ファイルは、データベース・サーバと同じ物理コンピュータに保存してください。または SAN や iSCSI 設定でアクセスできるようにしてください。リモート・ネットワーク・ディレクトリにデータベース・ファイルやトランザクション・ログ・ファイルを配置すると、パフォーマンスが低下したり、データが破壊されたり、サーバが不安定になったりする可能性があります。
詳細については、http://www.ianywhere.jp/developers/technotes/asa_db_file_stored_remotely.html を参照してください。

変更がディスクに書き込まれるとき

データベース・ファイルと同様に、トランザクション・ログは「**ページ**」、つまり固定サイズのメモリ領域に保持されます。トランザクション・ログに変更が記録されるとき、その内容はメモリ内のページに書き込まれます。変更は、次の状況のいずれかが発生した段階で、強制的にディスクに書き込まれます。

- ページが満杯になったとき
- COMMIT が実行されたとき

この方法によって、完了したトランザクションが確実にディスクに格納されるとともに、操作のたびにディスクに書き込む必要がないため、パフォーマンスも向上します。

設定オプションを使用すると、詳しい知識があるユーザはトランザクション・ログの動作を細かくチューニングできます。「cooperative_commits オプション [データベース]」 439 ページと「delayed_commits オプション [データベース]」 446 ページを参照してください。

トランザクション・ログ・ミラー

「**トランザクション・ログ・ミラー**」はトランザクション・ログの完全なコピーであり、トランザクション・ログと同時に管理されます。データベースが、ミラーされたトランザクション・ロ

グを持っている場合、データベースに対する変更は、トランザクション・ログとトランザクション・ログ・ミラーの両方に書き込まれます。デフォルトでは、データベースは、トランザクション・ログ・ミラーを持ちません。

トランザクション・ログ・ミラーは、重要なデータを二重に保護します。トランザクション・ログ・ミラーによって、トランザクション・ログでメディア障害が発生した場合に完全なデータ・リカバリが可能です。また、ミラーされたトランザクション・ログがあると、データベースの開始時に、データベース・サーバによるトランザクション・ログの自動検証が可能になります。「トランザクション・ログのメディア障害からの保護」 830 ページを参照してください。

メディア障害からのデータの保護

バックアップによって、メディア障害からデータを保護できます。

メディア障害からリカバリする実際の方法は、発生場所 (データベース・ファイルまたはトランザクション・ログ・ファイル) によって異なります。

データベース・ファイルで発生したメディア障害　データベース・ファイルが使用できなくなっても、トランザクション・ログが使用でき、所定の手順どおりに正しくバックアップしているかぎり、データベースにコミットされた変更はすべてリカバリできます。データベース・ファイルのコピーを最後にバックアップしてからの情報は、すべてバックアップされたトランザクション・ログまたはオンラインのトランザクション・ログに格納されています。

トランザクション・ログ・ファイルで発生したメディア障害　ミラーされたトランザクション・ログを使用しないかぎり、データベース・チェックポイントを最後に実行してからトランザクション・ログでメディア障害が発生するまでに入力された情報はリカバリできません。このため、SQL Remote 統合データベースなどの設定では、ミラーされたトランザクション・ログを使用することをおすすめします。これらの設定では、トランザクション・ログがなくなると、重要な情報が損失したり、レプリケーション・システムが破損したりする可能性があります。

参照

- 「障害からのデータの保護」 814 ページ
- 「データベース・ファイルで発生したメディア障害からの保護」 830 ページ
- 「トランザクション・ログのメディア障害からの保護」 830 ページ

バックアップ・プロシージャの設計

バックアップを作成する場合、トランザクション・ログを管理する方法について一連の選択肢があります。どの選択肢が適切であるかは、次の一連の要因によって異なります。

- データベースがレプリケーションに関連するかどうか。

  この章では、レプリケーションとは、SQL Remote レプリケーション、dbmlsync が実行されている Mobile Link 同期、または Replication Agent を使用しているデータベースを意味します。これらのレプリケーションの方法では、トランザクション・ログ、場合によっては古いトランザクション・ログへのアクセスが必要になります。

- トランザクション・ログ・ファイルのサイズが、使用可能なディスク領域に対してどれくらいの速さで拡大するか。トランザクション・ログの拡大が急速である場合、トランザクション・ログを使用可能な状態にしておくことが割に合わないことがあります。

バックアップの種類

この項は、バックアップの基本概念を理解している方を対象としています。

バックアップは、次のような複数の方法に分類されます。

- **フル・バックアップとインクリメンタル・バックアップ**　「**フル・バックアップ**」では、データベース・ファイルとトランザクション・ログの両方をバックアップします。「フル・バックアップの実行」 845 ページを参照してください。

  「**インクリメンタル・バックアップ**」では、トランザクション・ログだけをバックアップします。通常、フル・バックアップとフル・バックアップの間にインクリメンタル・バックアップを複数回実行します。「インクリメンタル・バックアップの実行」 846 ページを参照してください。

- **サーバ側のバックアップとクライアント側のバックアップ**
  サーバ側でバックアップを実行するには、BACKUP 文を実行するか、dbbackup -s オプションを使用します。これで、データベース・サーバでのバックアップが実行されます。BACKUP 文は SQL 文なので、サーバ側のバックアップをアプリケーションに簡単に組み込むことができます。また、クライアント／サーバ通信システムを介してデータを転送する必要がないので、通常、サーバ側のバックアップは高速です。「BACKUP 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  バックアップ・ユーティリティを使用して、クライアント・コンピュータからオンライン・バックアップを実行できます。

  次の項を参照してください。

  - 「バックアップ・ユーティリティ (dbbackup)」 640 ページ
  - 「イメージ・バックアップの作成」 821 ページ
  - 「DBBackup 関数」 『SQL Anywhere サーバ - プログラミング』

- **アーカイブ・バックアップとイメージ・バックアップ**
  「**アーカイブ・バックアップ**」では、データベース・ファイルとトランザクション・ログを 1 つのアーカイブ・ファイル (通常はテープ・ドライブ上) にコピーします。「**イメージ・バックアップ**」では、データベース・ファイルとトランザクション・ログ (任意) のコピーをそれぞれ別のファイルとして作成します。アーカイブ・バックアップはサーバ側のバックアップとしてだけ実行でき、フル・バックアップだけを作成できます。

  テープに直接バックアップする場合は、アーカイブ・バックアップを使用してください。それ以外の場合は、イメージ・バックアップを行った方が柔軟にトランザクション・ログ・ファイルを管理できます。「データベースを直接テープにバックアップする」 855 ページを参照してください。

- **オンライン・バックアップとオフライン・バックアップ**
  実行中のデータベースをバックアップすると、他のユーザによってデータベースが変更されている途中ではあっても、一貫したデータベースのスナップショットを得ることができます。オフライン・バックアップは単なるファイルのコピーです。オフライン・バックアップは、データベースが稼働中でなく、データベース・サーバが正しく停止された場合にだけ実行するようにしてください。

  この章では、オンライン・バックアップを中心に説明します。

- **ライブ・バックアップ**　コンピュータ全体に及ぶ障害からデータベースを保護するのに役立つ*継続的な*バックアップです。

  ライブ・バックアップを使用すると、トランザクション・ログの冗長コピーを作成できます。作成したコピーは、データベース・サーバを実行している第 1 システムが使用できなくなった場合に、第 2 システムの再起動に使用されます。ライブ・バックアップは継続的に実行され、サーバが停止した場合にのみ中止されます。システム障害が発生した場合は、バックアップされたトランザクション・ログを使って、システムをすばやく再起動できます。しかし、サーバが処理するロード量によってライブ・バックアップが遅れ、コミットされたすべてのトランザクションがバックアップされないことがあります。

  次の項を参照してください。

  - 「コンピュータ全体に及ぶ障害からの保護」 831 ページ
  - 「ライブ・バックアップの作成」 856 ページ

参照

- 「バックアップとリカバリの概要」 812 ページ
- 「バックアップの概要」 816 ページ

バックアップのスケジュール

バックアップのスケジュールでは、ほとんどの場合フル・バックアップを定期的に実行し、その間にトランザクション・ログのインクリメンタル・バックアップを行います。データのバックアップを作成する頻度について特に取り決めはありません。バックアップを作成する頻度は、データの重要性、データが更新または変更される頻度、その他の要因によって異なります。

ほとんどのバックアップ方式では、インクリメンタル・バックアップはこまめに行い、時々フル・バックアップを実行します。一般的には、週ごとにフル・バックアップを行い、1 日 1 回トランザクション・ログのインクリメンタル・バックアップを実行するというスケジュールから始めます。フル・バックアップとインクリメンタル・バックアップのどちらについても、オンライン (データベースの稼働中) またはオフラインで、サーバ側またはクライアント側で実行できます。アーカイブ・バックアップは、常にフル・バックアップです。

バックアップのスケジュールによってどの種類の障害からデータを保護できるかは、バックアップの頻度だけでなく、データベース・サーバの操作方法によっても異なります。「データ保護を目的としたデータベースの構成」 830 ページを参照してください。

常に複数のフル・バックアップを保管してください。*以前のバックアップに上書きする形でバックアップを作成すると、バックアップの最中にメディア障害が発生した場合、バックアップは失われてしまいます。*また、火事、洪水、地震、盗難、その他の破壊行為に備えて、フル・バックアップのコピーをオフサイトに保管してください。

SQL Anywhere のイベント・スケジュール機能を使用して、スケジュールした時刻に自動的にオンライン・バックアップを実行できます。「スケジュールとイベントの使用によるタスクの自動化」 869 ページを参照してください。

イメージ・バックアップの作成

イメージ・バックアップでは、データベース・ファイルとトランザクション・ログ (任意) のコピーがそれぞれ別のファイルとして作成されます。イメージ・バックアップを使用すると、アーカイブ・バックアップの場合よりトランザクション・ログを柔軟に管理できます。

イメージのバックアップの作成には、バックアップ・ユーティリティ、[バックアップ・イメージ作成] ウィザード、または BACKUP DATABASE 文を使用します。イメージ・バックアップは、サポートされるすべてのプラットフォームで使用できます。NetWare と Windows CE では、このバックアップがサポートされている唯一の種類です。

データベースをテープ・ドライブにバックアップする

イメージ・バックアップを使用してテープにバックアップするには、各バックアップ・コピーをとり、それをディスク・バックアップ・ユーティリティを使用してテープに格納する必要があります。

参照

- 「BACKUP 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「バックアップ・ユーティリティ (dbbackup)」 640 ページ
- 「RESTORE DATABASE 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「アーカイブ・バックアップの作成」 826 ページ

並列データベース・バックアップの知識

サーバ側のイメージ・バックアップをバックアップ・ユーティリティ (dbbackup) または BACKUP DATABASE 文を使用して実行する場合、データベースの並列バックアップが実行されます。並

列バックアップは、物理デバイスレベルの並列処理を使用して、バックアップ操作の完了に必要な総時間を節約します。並列バックアップは、Windows CE ではサポートされていません。

データベース・サーバは、データベース・ファイルが保存されている各ドライブについてリーダ・スレッドを作成します。ライタ・スレッドは、バックアップ・ディレクトリが存在するバックアップ先ドライブ用に作成されます。リーダとライタを別々に使用することで、I/O 処理が順次的にではなく並列に実行されます。

並列バックアップのパフォーマンスは、システムで最も低速なコンポーネントによって制限されます。ほとんどの場合、ボトルネックは物理ディスクですが、I/O コントローラやシステム・バスなど、その他すべてのコンポーネントがボトルネックになる可能性があります。それぞれのコンポーネントのデータ転送レートには上限があります。

BACKUP DATABASE 文とバックアップ・ユーティリティ (dbbackup) には、並列バックアップの動作を設定できるよう次のオプションが用意されています。

- チェックポイント・ログをコピーするタイミングと方法
- データベース・サーバから dbbackup へのデータ転送時に使用できる最大ページ数 (dbbackup の使用時のみ使用可)
- ライタの追加 (BACKUP 文の場合のみ)

バックアップは、必ず別の物理ドライブに作成してください。これにより、I/O 並列処理によりパフォーマンスが向上するだけでなく、ハードウェア障害が発生したときのデータの安全性が向上します。

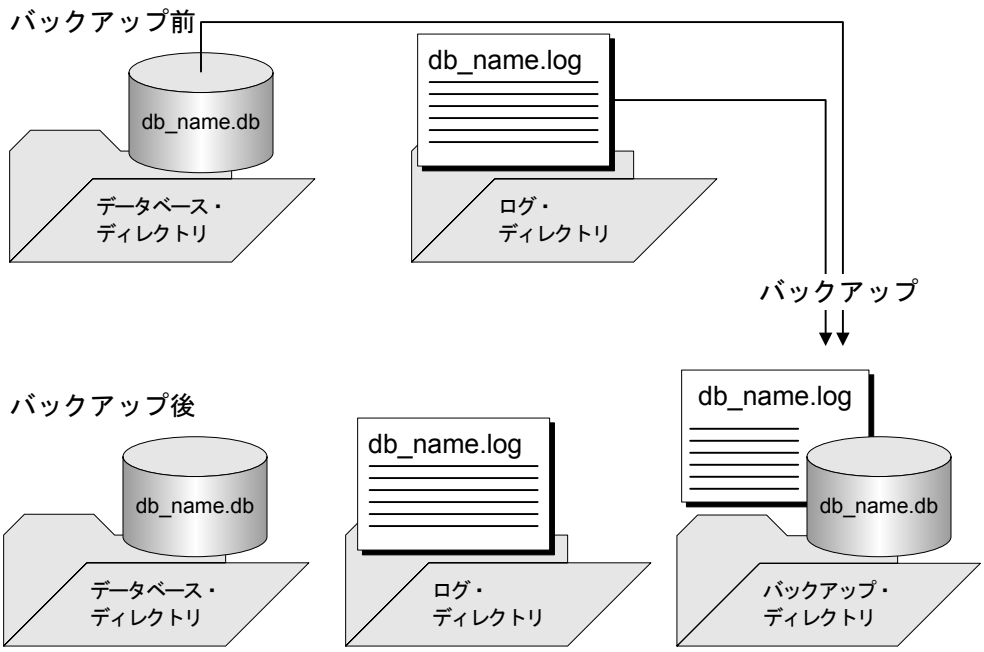
参照

- 「BACKUP 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「バックアップ・ユーティリティ (dbbackup)」 640 ページ

ディスク領域が十分な場合のバックアップ・スキーマ

運用コンピュータ (データベース・サーバが稼働しているコンピュータ) のディスク領域に余裕がある場合は、トランザクション・ログ・ファイルまたはチェックポイント・ログを管理するために特別なオプションを選択する必要はありません。この場合、イメージ・バックアップを使用できます。最も簡単なイメージ・バックアップでは、データベース・ファイルとトランザクション・ログのコピーが作成され、トランザクション・ログは所定の場所に格納されたままになります。すべてのバックアップにおいて、データベース・ファイルは所定の場所に格納されたままです。

この種類のフル・バックアップを次の図で説明します。インクリメンタル・バックアップでは、トランザクション・ログだけがバックアップされます。

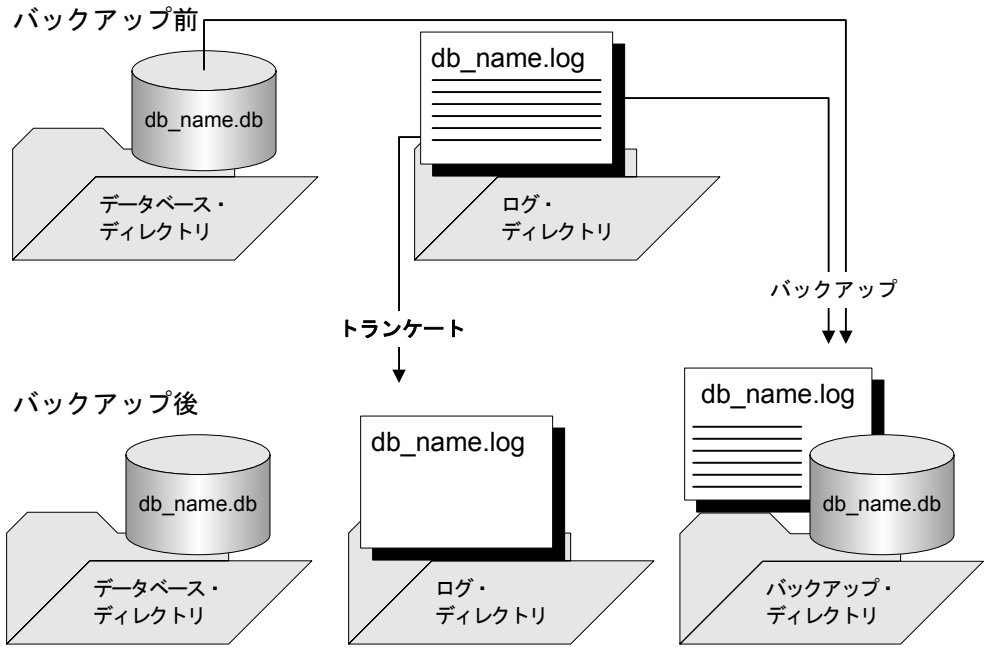


フル・バックアップの詳細については、「バックアップを作成し、元のトランザクション・ログを継続して使用する」 849 ページを参照してください。

レプリケーションに関連しないデータベースのバックアップ・スキーマ

多くの状況では、ディスク領域は制限されており、トランザクション・ログを無制限に増大させることは現実的ではありません。このような場合、バックアップが完了したらトランザクション・ログの内容を削除するように選択でき、それによってディスク領域を解放できます。ただし、データベースがレプリケーションに関連する場合は、レプリケーションではトランザクション・ログへのアクセスを必要とするので、このオプションを選択しないでください。

次の図で、ログ・ファイルをトランケートするフル・バックアップを説明します。インクリメンタル・バックアップでは、トランザクション・ログだけがバックアップされます。



各インクリメンタル・バックアップの後にトランザクション・ログを削除すると、最後のフル・バックアップが実行されてから複数の種類のトランザクション・ログが発生することがあるため、データベース・ファイル上のメディア障害からのリカバリが複雑になります。データベースを最新の状態にリカバリするためには、各トランザクション・ログを順番に適用する必要があります。

Mobile Link 統合データベースとして稼働するデータベースでは、この種類のバックアップを実行できます。これは、Mobile Link サーバがトランザクション・ログに依存しないためです。SQL Remote または Mobile Link *dbmlsync.exe* クライアント・アプリケーションを実行している場合は、古いトランザクション・ログを保存するために、適切なスキームを使用してください。

次の項を参照してください。

- 「レプリケーションに関連するデータベースのバックアップ・スキーマ」 824 ページ
- 「バックアップを作成し、元のトランザクション・ログを削除する」 850 ページ

レプリケーションに関連するデータベースのバックアップ・スキーマ

データベースが SQL Remote インストール環境の一部である場合、Message Agent は古いトランザクションにアクセスする必要があります。統合データベースの場合、データベース内に SQL Remote インストール環境全体のマスタ・コピーが格納されるので、完全なバックアップ手順が必要です。

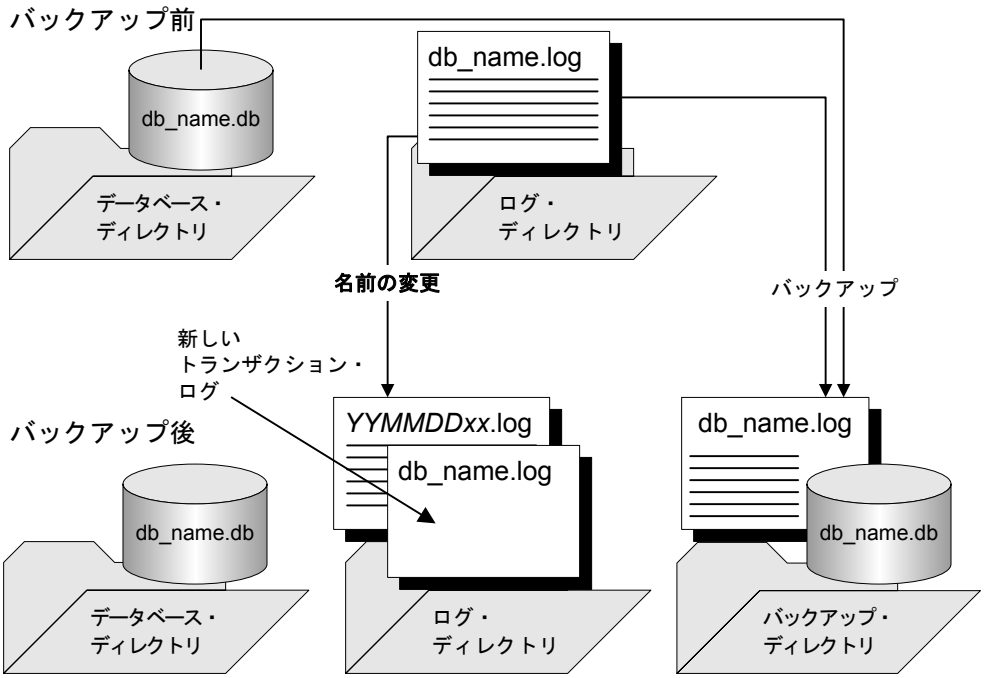
データベースが Replication Server インストール環境のプライマリ・サイトにある場合、Replication Agent は古いトランザクションにアクセスする必要があります。しかし、多くの場合ディスク領

域は制限されており、トランザクション・ログを無制限に増大させることは現実的ではありません。

データベースが Mobile Link の設定に関連していて、dbmlsync を使用している場合は、同じことに注意する必要があります。ただし、データベースが Mobile Link 統合データベースである場合は、古いトランザクション・ログは必要なく、前の項で説明したように、レプリケーションに関連しないデータベースのスキーマを使用できます。

このような場合には、トランザクション・ログの名前を変更してトランザクション・ログを再起動するようにバックアップ・オプションを選択できます。この種類のバックアップを実行すると、Message Agent と Replication Agent のために古いトランザクションの情報を保ちつつ、トランザクション・ログが無限に拡大するのを防ぐことができます。

この種類のバックアップを次の図で説明します。



この種のバックアップの実行方法については、「バックアップを作成し、元のトランザクション・ログの名前を変更する」 852 ページを参照してください。

オフライン・トランザクション・ログ

バックアップ操作では、トランザクション・ログのバックアップに加え、オンライン・トランザクション・ログのファイル名を *YYMMDDxx.log* という形式に変更します。このファイルはデータベース・サーバでは使用されませんが、Message Agent や Replication Agent では利用できます。これを「**オフライン**・トランザクション・ログ」と呼びます。新しいオンライン・トランザクション・ログの最初の名前には、古いオンライン・トランザクション・ログの名前が付けられます。

YYMMDDxx.log というファイル名は、順序付けではなく、識別だけを目的として使用されます。たとえば、最初のバックアップが 2000 年 12 月 10 日である場合、ログ・ファイル名は *001210AA.log* になります。最初の 2 桁は年、次の 2 桁は月、その次の 2 桁は日付を示し、最後の 2 文字によって、同じ日に実行された複数のバックアップを識別します。

Message Agent と Replication Agent は、オフライン・コピーを使用して古いトランザクションを必要に応じて提供できます。delete_old_logs データベース・オプションを On に設定すると、Message Agent と Replication Agent が必要としなくなった古いオフライン・ファイルは削除されるので、ディスク領域を節約できます。

レプリケーション・インストールにおけるリモート・データベースのバックアップ方法

リモート・データベースでは、バックアップ手順は統合データベースの場合ほど重要ではありません。データのバックアップを、統合データベースへのレプリケーションに頼る方法もあります。メディア障害が発生した場合は、統合データベースからリモート・データベースを再抽出しなければならず、レプリケートされていないオペレーションは失われます (ログ変換ユーティリティを使用して、失われたオペレーションのリカバリを実行することは可能です。「ログ変換ユーティリティ (dbtran)」 688 ページを参照してください)。

レプリケーションに頼ってリモート・データベースのデータを保護する場合でも、トランザクション・ログが大きくなりすぎるのを防ぐために、リモート・データベースでバックアップを定期的に実行する必要があります。統合データベースで使用するのと同じオプション (ログの名前変更と再起動) を使用して Message Agent を実行し、Message Agent が名前変更されたログ・ファイルにアクセスできるようにします。リモート・データベースで delete_old_logs オプションを On に設定すると、不要になった古いログ・ファイルが Message Agent によって自動的に削除されます。

自動的にトランザクション・ログの名前を変更

-x Message Agent オプションを使用すると、データベース・サーバを停止した際に、リモート・コンピュータでトランザクション・ログの名前を変更する必要がなくなります。-x オプションは、トランザクション・ログが出力メッセージ用にスキャンされた後で、トランザクション・ログの名前を変更します。

アーカイブ・バックアップの作成

アーカイブ・バックアップは、メイン・データベース・ファイル、トランザクション・ログ、すべての追加 DB 領域など、バックアップに必要なすべての情報が含まれた単一のファイルです。アーカイブ・バックアップはサーバ側のバックアップとしてだけ実行でき、フル・バックアップだけを作成できます。アーカイブ・バックアップは、ファイルまたはテープ・ドライブに保存できます。アーカイブ・バックアップの作成には、BACKUP DATABASE 文または Sybase Central の [データベース・バックアップ] ウィザードを使用します。

アーカイブ・バックアップからのデータベースのリストアには、Sybase Central の [データベース・リストア] ウィザードまたは RESTORE DATABASE 文を使用します。

アーカイブ・バックアップは、Windows と UNIX の各プラットフォームでのみサポートされています。Windows CE では、イメージ・バックアップのみが許可されています。

データベースをテープ・ドライブにバックアップする

アーカイブ・バックアップを使用すると、テープ・ドライブに直接バックアップできます。アーカイブ・バックアップは、常にフル・バックアップです。アーカイブ・バックアップでは、データベース・ファイルとトランザクション・ログのコピーを作成しますが、これらのコピーは 1 つのファイルに格納されます。

参照

- 「BACKUP 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「RESTORE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「アーカイブ・バックアップのリストア」 860 ページ
- 「イメージ・バックアップの作成」 821 ページ

バックアップとリカバリのプランの設計

データ保護の信頼性を高めるには、バックアップのスケジュールを立て、実行してください。検証済みのリカバリ手順指示書を用意しておく必要もあります。

一般的なスケジュールとしては、定期的にフル・バックアップを実行し、その間に数回インクリメンタル・バックアップを行う方法があります。各バックアップの頻度は、保護するデータの種類によって異なります。

内部バックアップを使用する場合は、SQL Anywhere のスケジュール機能を使用して自動的にバックアップできます。スケジュールを指定しておくと、以後データベース・サーバによってバックアップが自動的に実行されます。「スケジュールとイベントの使用によるタスクの自動化」 869 ページを参照してください。

データベースを使用する組織がどれだけの時間データベース内のデータにアクセスしなくても支障がないかによって、リカバリに割り当てられる時間の上限が決定付けられます。バックアップとリカバリのプランは、この条件に合うように開発し、テストしてください。

データベース・ファイルとトランザクション・ログ・ファイルのメディア障害に対して、必要な保護措置が取られていることを確認してください。レプリケーション環境で作業している場合は、ミラーされたトランザクション・ログの使用を検討してください。「メディア障害からのデータの保護」 818 ページを参照してください。

リカバリ時間に影響する要因

リカバリの所要時間は、使用可能なハードウェア、データベース・ファイルのサイズ、リカバリに使用する媒体、ディスク領域、予期しないエラーなどの外部要因の影響を受けます。バックアップ方式を計画するときには、リカバリ・コマンドの入力、テープの検索やロードといった、その他のタスクにかかる時間も考慮してリカバリの所要時間を決めてください。

リカバリの手順に関係するファイルが増えれば、それだけリカバリが失敗する要因も増えます。バックアップとリカバリの方式を開発していく過程では、リカバリ・プランの再検討を忘れないようにしてください。「バックアップとリカバリ処理の設定」 845 ページを参照してください。

データベースの妥当性の確認

データベース・ファイルの破損は、データベース・サーバがデータベース内の破損部分にアクセスするまで判明しないことがあります。データ保護処理の一部として、データベースにエラーがないことを定期的にチェックしてください。チェックするには、Sybase Central の [データベース検証] ウィザードや、検証ユーティリティ (dbvalid) などのツールを使用して、データベースを「**検証**」します。データベースの検証は、バックアップの前後両方に実行してください。検証を実行するには、VALIDATE パーミッションが必要です。

検証では、全テーブル内の全ローのスキャンと、テーブルの各インデックスの各ローの検索が実行されます。そのため、検証には各テーブルに対する排他的なアクセスが必要です。このことを考慮すれば、データベース上に他のアクティビティがない場合に、データベースの検証を行うのが最適です。-fx オプションを使用してエクスプレス検証を実行した場合、データベースの検証においてデータ、連続したローの構造、外部キー関係は検証されません。

バックアップが作成されているときに実行中のトランザクションがなかったという確信が持てる場合は、データベース・サーバによるリカバリの手順を実行する必要はありません。代わりに、読み込み専用のデータベース・オプションを使用して、バックアップしたデータベースに妥当性検査を実行できます。「-r サーバ・オプション」 189 ページを参照してください。

ヒント
WAIT BEFORE START 句を使用して BACKUP 文を実行すると、トランザクションの処理中にはバックアップが開始されません。

データベース・ファイル内のベース・テーブルが破損している場合は、メディア障害として対処し、前のバックアップからリカバリしてください。インデックスが破損している場合は、インデックスなしでデータベースをアンロードして、再ロードします。

警告
データベースとトランザクション・ログのバックアップ・コピーには、どのような変更でも加えるべきではありません。バックアップ中に処理中のトランザクションがなかった場合、または BACKUP DATABASE WITH CHECKPOINT LOG RECOVER か WITH CHECKPOINT LOG NO COPY を指定した場合は、バックアップ・データベースの妥当性を読み込み専用モードでチェックできます。一方、トランザクションの処理中だった場合、または BACKUP DATABASE WITH CHECKPOINT LOG COPY を指定した場合は、検証の開始時にデータベース・サーバがデータベースのリカバリを実行する必要が生じます。リカバリを実行するとバックアップ・コピーに変更が加えられますが、これは望ましいことではありません。

チェックサムの検証

チェックサムを有効にしてデータベースを作成した場合、ディスク・ページの妥当性を確認できます。チェックサムの検証には、DBA または VALIDATE 権限が必要です。

チェックサムを有効にしたデータベースの場合、チェックサムは各データベース・ページで計算され、ページがディスクに書き込まれる際にその値が格納されます。検証ユーティリティ (dbvalid) または Sybase Central の [データベース検証] ウィザードを使用してチェックサムの検証を実行できます。この検証は、ディスクからのデータベース・ページの読み込みとそのページに対する

チェックサムの計算で構成されています。計算されたチェックサムが格納されているページのチェックサムと一致しない場合、ディスク上で、または書き込み中に、そのページが変更または破壊されています。1 つまたは複数のページが破壊されている場合、エラーが返されて無効なページに関する情報がサーバ・メッセージ・ウィンドウに表示されます。

チェックサム検証の詳細については、「VALIDATE 文」『SQL Anywhere サーバ - SQL リファレンス』と「検証ユーティリティ (dbvalid)」 756 ページを参照してください。

注意
チェックサムが有効かどうかに関係なく、データベース・サーバはすべてのデータベース内の重要なデータベース・ページのチェックサムを計算します。計算されたチェックサムは、オフラインの破損を検出するために使用します。チェックサムによって、重要なページの破損が原因で他のデータが破損するのを防ぐことができます。データベース・サーバがチェックサムを計算するので、チェックサムが有効に指定されていないデータベースが壊れた場合、データベース・サーバは致命的なエラーとともに停止します。
また、チェックサムが有効に指定されていないデータベースを検証した場合でも、そのデータベースに重要なページの破損があると、dbvalid はチェックサム検証に関する警告を返します。

参照

- 「データベースの検証」 847 ページ
- 「トランザクション・ログの検証」 848 ページ
- 「単一テーブルの検証」 848 ページ
- 「VALIDATE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「データベース検証時のパフォーマンスの改善」 843 ページ

データ保護を目的としたデータベースの構成

パフォーマンスを低下させることなくメディア障害から保護できるように、データベースとデータベース・サーバを構成する方法が複数あります。

データベース・ファイルで発生したメディア障害からの保護

デフォルトでは、データベースを作成すると、トランザクション・ログはデータベースと同じデバイス上の同じディレクトリに格納されます。この方法ではすべての種類のメディア障害からデータベースを保護できないので、運用上、トランザクション・ログを別の場所に格納してください。

メディア障害から確実に保護するには、データベース・ファイルと異なるデバイスにトランザクション・ログを保存してください。複数のハード・ドライブがあるコンピュータの中には、実際には 1 つの物理ストレージ・ドライブをいくつかの論理ドライブまたはパーティションに区切っただけのものがあります。メディア障害に確実に備えるには、最低 2 つの物理ストレージ・デバイスがあるコンピュータを使用します。

トランザクション・ログを別のデバイスに格納すると、ディスクのヘッドが、トランザクション・ログとメイン・データベース・ファイルの間を移動しなくて済むことからも、パフォーマンスの改善が期待できます。

ネットワーク・ディレクトリにトランザクション・ログを格納しないでください。ネットワークを介してページに対する読み取りと書き込みを行うと、パフォーマンスの低下やファイルの破損を招く可能性があります。

参照

- 「データベースの作成」 『SQL Anywhere サーバ - SQL の使用法』
- 「トランザクション・ログの場所の変更」 866 ページ

トランザクション・ログのメディア障害からの保護

大容量または非常に重要なアプリケーションを実行する場合は、トランザクション・ログ・ミラーを使用することをおすすめします。たとえば、SQL Remote 設定における統合データベースでは、レプリケーションはトランザクション・ログに依存します。トランザクション・ログが損傷した場合、データのレプリケーションは失敗します。

ミラーされたトランザクション・ログを使用している場合は、ログのいずれかに書き込もうとしてエラーが発生すると (ディスクが満杯の場合など)、データベース・サーバが停止します。トランザクション・ログ・ミラーの目的は、いずれかのログ・デバイスでメディア障害が発生したときに完全にリカバリできるようにすることです。1 つのログを使用してサーバを実行し続けると、この目的は達成できません。

データベース・サーバの起動時に -fc オプションを指定して、データベース・サーバでファイル・システム・フル条件が発生した場合のコールバック関数を実装できます。「-fc サーバ・オプション」 165 ページを参照してください。

トランザクション・ログ・ミラーの保存先

ログのミラーリングを行うと、各データベース・ログへの書き込みを 2 回ずつ行う必要があるため、パフォーマンスが低下します。低下の程度は、データベース内のデータ転送の形態と量、データベースとログの物理的な設定の方法によって異なります。

トランザクション・ログ・ミラーは、トランザクション・ログとは別のデバイスに保管してください。そうすることによって、パフォーマンスが向上します。また、一方のデバイスが故障しても、ログのもう一方のコピーにリカバリのためのデータが残ります。

トランザクション・ログ・ミラーに代わる方法

ミラーされたトランザクション・ログに代わる方法として、ディスク・コントローラによるハードウェア・ミラーリング、または Windows と NetWare に用意されているオペレーティング・システム・レベルのソフトウェア・ミラーリングを使用します。通常、ハードウェア・ミラーリングは費用がかかりますが、パフォーマンスは向上します。

ライブ・バックアップを使用すると、トランザクション・ログ・ミラーに似た方法でデータを保護することができます。「ライブ・バックアップとトランザクション・ログ・ミラーの違い」 832 ページを参照してください。

ミラーされたトランザクション・ログでデータベースを作成する方法については、「初期化ユーティリティ (dbinit)」 662 ページを参照してください。

既存のデータベースでミラーされたトランザクション・ログを使うように変更する方法については、「トランザクション・ログ・ユーティリティ (dblog)」 735 ページを参照してください。

コンピュータ全体に及ぶ障害からの保護

「**ライブ・バックアップ**」を使用してトランザクション・ログの重複コピーを作成すると、データベース・サーバを実行するコンピュータが使用できなくなった場合に、そのコピーを使用してセカンダリ・コンピュータでシステムを再起動できます。

ライブ・バックアップは継続的に実行され、サーバが停止した場合にのみ中止されます。システム障害が発生した場合は、バックアップされたトランザクション・ログを使って、システムをすばやく再起動できます。しかし、サーバが処理するロード量によってライブ・バックアップが遅れ、コミットされたすべてのトランザクションがバックアップされないことがあります。

ライブ・バックアップの代わりにデータベース・ミラーリングを使用することもできます。「データベース・ミラーリングの概要」 888 ページを参照してください。

参照

- 「ライブ・バックアップの作成」 856 ページ
- 「ライブ・バックアップのリカバリ」 860 ページ

ライブ・バックアップとトランザクション・ログ・ミラーの違い

ライブ・バックアップとトランザクション・ログ・ミラーは両方とも、トランザクション・ログのセカンダリ・コピーを作成します。しかし、ライブ・バックアップの使用とトランザクション・ログ・ミラーの使用には、次のようないくつかの違いがあります。

- **通常、ライブ・バックアップは別のコンピュータで実行される**　別のコンピュータでバックアップ・ユーティリティを実行すると、バックアップ・ログ・ファイルへの書き込みはデータベース・サーバによっては行われません。また、データ転送は SQL Anywhere クライアント／サーバ通信システムによって実行されます。したがって、パフォーマンスへの影響を低減でき、信頼性も向上します。

  トランザクション・ログ・ミラーの場合、別のコンピュータで実行することはおすすめしません。実行するとパフォーマンスに問題が発生したりデータが破壊されたりする可能性があり、コンピュータ間の接続に障害が発生すると、データベース・サーバが停止します。

- **ライブ・バックアップは、コンピュータが使用できなくなる状況から保護する**　トランザクション・ログ・ミラーを別のデバイスに保存しても、コンピュータ全体が使用できなくなってしまえば、リカバリには時間がかかります。2 台のコンピュータで一連のディスクへのアクセスを共有するように構成するのも 1 つの方法です。

- **ライブ・バックアップはデータベース・サーバより処理が遅れることがある**　ミラーされたトランザクション・ログの場合は、コミットされたトランザクションを完全にリカバリするのに必要な情報がすべて含まれます。ライブ・バックアップの場合、サーバが処理するロード量によっては処理が遅れ、コミットされたすべてのトランザクションがバックアップされないことがあります。

ライブ・バックアップと定期的なバックアップ

トランザクション・ログのライブ・バックアップの長さは、アクティブなトランザクション・ログと常に同じか短くなります。ライブ・バックアップの実行中に、別のバックアップがトランザクション・ログを再起動すると (dbbackup -r または dbbackup -x)、ライブ・バックアップは自動的にライブ・バックアップ・ログをトランケートして、新しいトランザクション・ログの最初からライブ・バックアップを再起動します。「ライブ・バックアップの作成」 856 ページを参照してください。

トランザクション・ログ・サイズの制御

どの種類のバックアップが最適かは、トランザクション・ログのサイズによって決まります。また、このサイズはリカバリの所要時間にも影響します。

すべてのテーブルに簡単なプライマリ・キーを設定することによって、トランザクション・ログ・ファイルの増大する速度を制御できます。プライマリ・キーまたは NULL 入力不可のユニーク・インデックスがないテーブルで更新または削除を実行すると、対象ローの内容がすべてトランザクション・ログに保存されます。プライマリ・キーが定義されている場合は、データベース・サーバはそのカラム値を保存するだけでローをユニークに識別できます。テーブルのカラム数が多い場合、またはテーブルのカラムに長いデータを含む場合は、プライマリ・キーが定義されていないと、トランザクション・ログのページがすぐに満杯になります。このように、データ

の余分な書き込みによって、ディスク領域を多く必要とするだけではなく、パフォーマンスが低下します。

プライマリ・キーがない場合、サーバはテーブル上で UNIQUE NOT NULL インデックス (または UNIQUE 制約) を探します。NULL 値を許容する UNIQUE インデックスでは十分に識別できないためです。

バックアップとリカバリの内部

この項では、バックアップ中と、システム障害からの自動リカバリ中に使用される内部メカニズムについて説明します。

バックアップの内部メカニズム

バックアップを実行するとき、多くのユーザがデータベースを使用中である可能性があります。バックアップしたデータベースを後でリストアする必要がある場合は、どの情報がバックアップされて、どの情報がバックアップされていないかを把握しておく必要があります。

データベース・サーバでは、次の順序でバックアップを実行します。

1. チェックポイントを発行します。バックアップが完了するまで、これ以上チェックポイントは使用できません。
2. バックアップ命令がフル・バックアップを実行する場合は、データベース・ファイルのバックアップを作成します。
3. トランザクション・ログのバックアップを作成します。

   ログの最後のページが読み込まれる前に、トランザクション・ログに記録されたすべてのオペレーションがバックアップされます。バックアップされるオペレーションには、バックアップ命令の発行後に発行された命令も含まれます。

   通常、トランザクション・ログのバックアップ・コピーは、オンライン・トランザクション・ログよりも小さくなります。データベース・サーバでは、オンライン・トランザクション・ログに 64 KB 単位で領域を割り当てるので、トランザクション・ログ・ファイルのサイズには空のページも含まれるのが一般的です。しかし、空でないページだけがバックアップされます。
4. バックアップ命令でトランザクション・ログをトランケートするか名前を変更する必要がある場合、コミットされていないトランザクションは新しいトランザクション・ログに持ち越されます。

   トランザクション・ログの名前変更とトランケーションについては、「バックアップ・プロシージャの設計」 819 ページを参照してください。
5. データベースのバックアップ・イメージにマークを付けて、リカバリが必要であることを示します。こうすると、バックアップの開始以降に実行されたオペレーションはすべてリカバリされます。また、チェックポイントの時点で完了していなかったオペレーションは、コミットされていなければ取り消されます。

バックアップとリカバリの制限

データベース・サーバでは、バックアップ処理中に次のオペレーションを実行できないようにします。

- 別のバックアップ (ライブ・バックアップを除く)
- チェックポイント (バックアップ命令自体によって発生するチェックポイントは除く)
- チェックポイントを発生させる文。これには、LOAD TABLE 文と TRUNCATE TABLE 文の他にデータ定義文も含まれます。

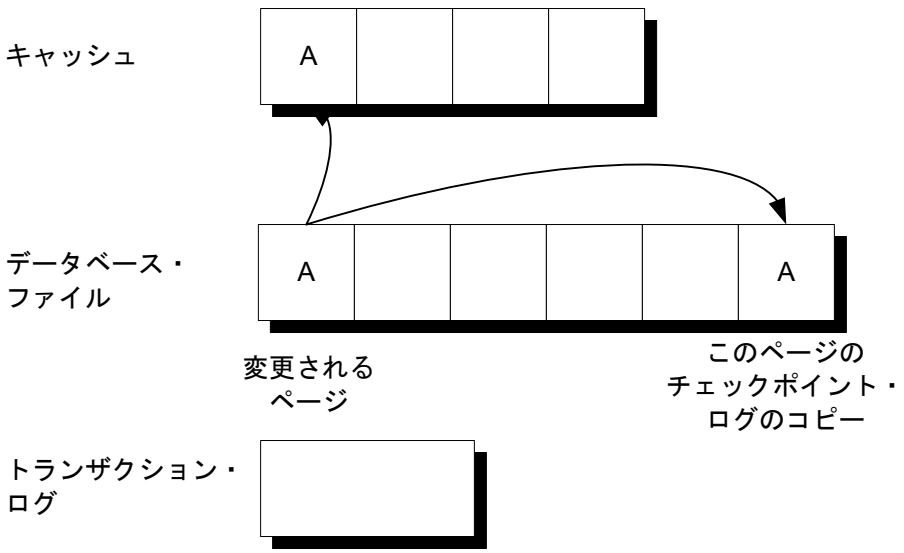
リカバリ (バックアップのリストアを含む) の間、データベースの別のユーザに許可されているアクションはありません。

チェックポイントとチェックポイント・ログ

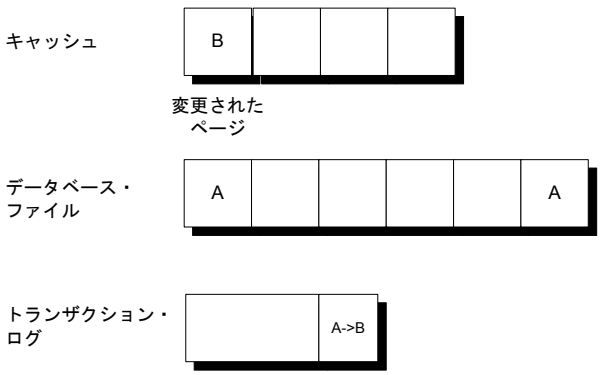
データベース・ファイルはページで構成されています。ページとは、ハード・ディスク内の決められたサイズの領域です。チェックポイント・ログは、データベース・ファイルの最後にあります。セッション中は必要に応じてチェックポイント・ログにページが追加され、セッションの最後にはチェックポイント・ログ全体が削除されます。

ページが更新される (「**ダーティ**」になる) 前に、データベース・サーバでは次のオペレーションが実行されます。

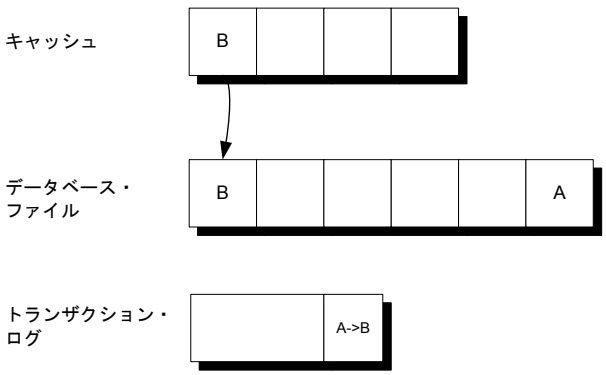
- ページがメモリに読み込まれ、データベース・キャッシュ内に格納されます。
- 元のページのコピーが作成されます。これらのコピーされたページを「**チェックポイント・ログ**」と呼びます。



ページに加えた変更は、キャッシュ内のコピーに適用されます。パフォーマンス上の理由から、ページはすぐにはディスクのデータベース・ファイルに書き込まれません。



キャッシュが満杯になると、変更されたページはディスクに書き込まれます。チェックポイント・ログのコピーは変更されません。



「**チェックポイント**」とは、ダーティ・ページがすべてディスクに書き込まれる時点のことで、ディスク上にあるデータベースの既知の一貫性のある状態を示します。チェックポイントの後で、チェックポイント・ログの内容が削除されます。空のチェックポイント・ログ・ページは、同一セッション中はチェックポイント・ログに存在し、新しいチェックポイント・ログ・データに再利用できます。チェックポイント・ログのサイズが大きくなると、データベース・ファイルも大きくなります。

チェックポイント時には、データベースの全データがディスクのデータベース・ファイルに格納されます。データベース・ファイルの情報は、トランザクション・ログの情報と一致します。リカバリ時には、データベースはまず最新のチェックポイントでリカバリされ、次にチェックポイント以降の変更内容が適用されます。

空のチェックポイント・ログ・ページもすべて含むチェックポイント・ログ全体は、セッションが終了するたびに削除されます。チェックポイント・ログを削除すると、データベースのサイズが小さくなります。

データベース・サーバは、チェックポイントを開始し、その実行中に他の操作を実行できます。ただし、チェックポイントがすでに進行中だった場合、ALTER TABLE や CREATE INDEX など、新しいチェックポイントを開始する操作はすべて、現在のチェックポイントの完了を待ちます。

チェックポイントがいつ実行されるかの詳細については、「データベース・サーバがチェックポイントのタイミングを決定する方法」 842 ページを参照してください。

トランザクションとロールバック・ログ

データベースの内容に加えられた変更は、「**ロールバック・ログ**」に記録されます。このログは、トランザクションがロールバックされた場合、またはシステム障害の発生時にトランザクションがコミットされていなかった場合に、変更をキャンセルするために使用されます。接続ごとに個別のロールバック・ログがあります。トランザクションのコミットまたはロールバックが行われると、その接続のロールバック・ログの内容は削除されます。ロールバック・ログは、データベースに格納されます。ロールバック・ログ・ページは、変更されるほかのページとともにチェックポイント・ログにコピーされます。

ロールバック・ログは「**取り消しログ**」とも呼ばれます。

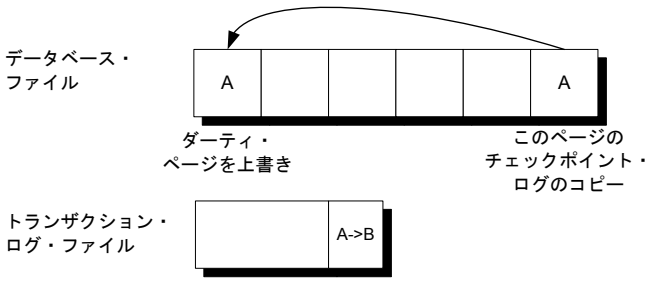
トランザクション処理の詳細については、「トランザクションと独立性レベル」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

自動リカバリ処理

正常なオペレーションでデータベースが停止すると、データベース・サーバでチェックポイントが実行され、データベース内のすべての情報がデータベース・ファイル内に格納されます。これは、「**クリーン**」な停止と呼ばれます。

データベースを起動するたびに、データベース・サーバは最後の停止がクリーンだったのか、システム障害の結果だったのかをチェックします。データベースの停止がクリーンでなかった場合は、システム障害からリカバリするために、次の手順が自動的に実行されます。

1. **最新のチェックポイントにリカバリする** チェックポイント・ログ・ページのコピーでチェックポイント以降に加えられた変更を上書きすることによって、すべてのページが最新のチェックポイント時の状態にリストアされます。


データベース・
ファイル
A
A
ダーティ・
ページを上書き
このページの
チェックポイント・
ログのコピー
トランザクション・
ログ・ファイル
A->B

2. **チェックポイント以降に加えられた変更を適用する** チェックポイントからシステム障害が発生するまでの間に加えられた変更が適用されます。この変更内容はトランザクション・ログに格納されています。
3. **コミットされていないトランザクションをロールバックする** コミットされていないトランザクションが、ロールバック・ログを使用してロールバックされます。

データベース・サーバがチェックポイントのタイミングを決定する方法

最後のチェックポイント以降の時間と作業量の増加に伴い、ダーティ・ページをディスクに書き込む優先度も増します。この優先度は、以下の要因によって決まります。

- **チェックポイントの緊急度** 最後のチェックポイント以降の経過時間を、データベースのチェックポイント時間の設定に対するパーセンテージで表したものです。チェックポイント間の最大時間は、サーバの -gc オプションを使って分単位で設定します。checkpoint_time オプションを使用して、希望の時間を設定することもできます。「checkpoint_time オプション [データベース]」 432 ページを参照してください。
- **リカバリの緊急度** データベースの障害が直ちに発生した場合のリカバリに必要な時間の推計です。システム障害が発生したときにリカバリを行う最大時間は、サーバの -gr オプションを使って分単位で設定します。recovery_time オプションを使用して、希望の時間を設定することもできます。「recovery_time オプション [データベース]」 488 ページを参照してください。

チェックポイントの緊急度とリカバリの緊急度のどちらも、サーバがダーティ・ページの書き込みを行うだけのアイドル時間を持てないときに重要です。

チェックポイントの頻度が高いとシステム障害からのリカバリは速くなります。しかし、データベース・エンジンがダーティ・ページを書き出す作業が増えます。

チェックポイントの頻度を制御するデータベース・オプションは 2 つあります。checkpoint_time は、チェックポイントどうしの最大期待間隔を制御し、recovery_time はシステム障害時の最大期待リカバリ時間を制御します。

データベースの他のアクティビティがあるためにダーティ・ページが 0 になり、緊急度が 50% を超えている場合、チェックポイントは自動的に発生します。

チェックポイントの緊急度とリカバリの緊急度の値は、チェックポイントが発生するまで増加を続け、チェックポイントが発生すると 0 に戻ります。チェックポイント発生以外の状況でこれらの値が小さくなることはありません。

参照

- 「-gc サーバ・オプション」 168 ページ
- 「-gr サーバ・オプション」 173 ページ

データベースの開始時にトランザクション・ログを検証する

データベース・サーバは、トランザクション・ログ・ミラーを使用するデータベースの起動時に一連の検証を行い、自動リカバリ操作を実行してトランザクション・ログとそのミラーが壊れていないかを確認します。壊れている場合は、いくつかの問題を修正します。

起動時には、トランザクション・ログとミラーを比較して、2 つのファイルが同一であるかを調べます。同一であれば、データベースは通常どおりに起動します。データベースの開始は、このログとミラーの比較のために時間がかかります。

システム障害のためにデータベースが停止した場合、操作の一部がトランザクション・ログには書き込まれても、ミラーには書き込まれていない可能性があります。トランザクション・ログとミラーのどちらか短い方のファイルを最後までチェックして 2 つのファイルが同じであることをサーバが確認すると、長い方のファイルの残りの部分が短い方のファイルにコピーされます。これによって、ログとミラーは同一になります。このリカバリ作業の後に、サーバは正常に起動します。

ファイルの短い部分の内容がログとミラーで異なっている場合は、どちらかのファイルが破損しています。この場合、データベースは起動しないで、エラー・メッセージが表示され、トランザクション・ログかミラーのどちらかが無効であることを知らせます。

データベース検証時のパフォーマンスの改善

大規模なデータベースを、テーブルとその最大インデックスを格納するのに十分なキャッシュがないサーバ上で実行している場合、そのデータベースに対して VALIDATE TABLE 文を実行すると時間がかかる場合があります。このような場合には、テーブルのすべてのページがインデックスごとに 1 回以上読み込まれることが多くなります。また、インデックス・ルックアップで全比較が必要な場合は、ページの読み込み回数がページ数ではなくテーブルのロー数に比例することがあります。

検証にかかる時間を短縮するには、VALIDATE TABLE 文で WITH EXPRESS CHECK オプションを使用するか、dbvalid ユーティリティで -fx オプションを使用します。データベースのサイズ、キャッシュのサイズ、使用する検証の種類によって異なりますが、これらの 2 つの機能によって、検証の実行にかかる時間を大幅に短縮できます。

エクスプレス検証では、テーブルの各ローが読み込まれ、すべてのカラムが評価されます。各インデックスは 1 回完全にスキャンされ、検査によってインデックスで参照されているローがテーブルに存在することが確認されます。エクスプレス・チェック・オプションでも、個々のインデックス・ページの妥当性が検査されます。テーブルのロー数はインデックスのエントリ数と同じである必要があります。エクスプレス・オプションで時間が短縮されるのは、各ローについて個々のインデックス・ルックアップを実行しないためです。

エクスプレス・チェック機能では個々のルックアップを実行しないため、エクスプレス検証機能を使用すると一部のインデックス破損を見逃してしまう可能性がまれにあります。インデックス破損が発生した場合でも、検証によってすべてのデータが読み込めることが確認されているため、データベースをアンロードし再構築することによってデータをリカバリできます。また、ALTER INDEX 文の REBUILD 句を使用してインデックス破損を修正することもできます。「ALTER INDEX 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

エクスプレス検証がサポートされているのは、Adaptive Server Anywhere 7.0 以降で作成したデータベースのみです。

- 「VALIDATE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「検証ユーティリティ (dbvalid)」 756 ページ
- 「sa_validate システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』

バックアップとリカバリの作業

この項では、バックアップとリカバリに関連する作業手順をまとめて説明します。

バックアップとリカバリ処理の設定

定期的にバックアップを実行すること、正しく動作することを確認済みのリカバリ用コマンドを準備しておくことは、総合的なバックアップとリカバリ設計の一部です。「バックアップとリカバリのプランの設計」 827 ページを参照してください。

- **バックアップとリカバリのプランを実行するには、次の手順に従います。**

1. バックアップとリカバリのコマンドを作成して検証します。
2. バックアップとリカバリのコマンド実行にかかる時間を測定します。
3. バックアップ・コマンドやバックアップの格納先を記述した手順書を作成し、使用している命名規則、実行するバックアップの種類なども明記しておきます。
4. 手順のとおりに運用サーバにバックアップを設定します。
5. 予期しないエラーを防止するために、バックアップ手順をモニタします。手順を変更した場合は、必ず手順書にも反映するようにします。

参照

- 「フル・バックアップの実行」 845 ページ
- 「インクリメンタル・バックアップの実行」 846 ページ

フル・バックアップの実行

フル・バックアップでは、データベース・ファイルとトランザクション・ログ・ファイルをバックアップします。

フル・バックアップとインクリメンタル・バックアップの違いについては、「バックアップの種類」 819 ページを参照してください。

- **フル・バックアップを作成するには、次の手順に従います (概要)。**

1. データベースに対して DBA 権限があることを確認します。
2. 妥当性検査を実行して、データベースが壊れていないことを確認します。妥当性検査には、検証ユーティリティまたは sa_validate ストアド・プロシージャを使用します。「データベースの検証」 847 ページを参照してください。
3. データベース・ファイルとトランザクション・ログのバックアップをとります。

   バックアップ操作の実行方法については、次の項を参照してください。

- 「バックアップを作成し、元のトランザクション・ログを継続して使用する」 849 ページ
- 「バックアップを作成し、元のトランザクション・ログを削除する」 850 ページ
- 「バックアップを作成し、元のトランザクション・ログの名前を変更する」 852 ページ

注意

妥当性検査では、データベースのテーブル全体に対する排他的なアクセスが必要です。

詳細と別の方法については、「データベースの妥当性の確認」 828 ページを参照してください。

データベースのバックアップ・コピーを検証する場合は、必ず読み込み専用モードで作業してください。-r オプションを使用してデータベース・サーバを起動すると、読み込み専用モードになります。

インクリメンタル・バックアップの実行

インクリメンタル・バックアップでは、トランザクション・ログ・ファイルだけがバックアップされます。各フル・バックアップ間に、できるだけ複数のインクリメンタル・バックアップを作成するようにしてください。

フル・バックアップとインクリメンタル・バックアップの違いについては、「バックアップの種類」 819 ページを参照してください。

♦ インクリメンタル・バックアップを作成するには、次の手順に従います (概要)。

1. データベースに対して DBA 権限があることを確認します。
2. データベース・ファイルではなく、トランザクション・ログだけのバックアップをとります。

バックアップ操作の実行方法の詳細については、次の項を参照してください。

- 「バックアップを作成し、元のトランザクション・ログを継続して使用する」 849 ページ
- 「バックアップを作成し、元のトランザクション・ログを削除する」 850 ページ
- 「バックアップを作成し、元のトランザクション・ログの名前を変更する」 852 ページ

注意

データベース・ファイルとトランザクション・ログ・ファイルのバックアップ・コピーには、オンライン・バージョンのファイルと同じ名前が付けられます。たとえば、サンプル・データベースのバックアップを作成する場合、バックアップのコピーは *demo.db* と *demo.log* という名前になります。バックアップ文を繰り返す場合は、バックアップ・コピーを上書きしないように新しいバックアップ・ディレクトリを選択してください。

トランザクション・ログのバックアップ・コピーの名前を変更して、インクリメンタル・バックアップ・コマンドを繰り返す方法については、「バックアップ中にトランザクション・ログのバックアップ・コピーの名前を変更する」 854 ページを参照してください。

データベースの検証

データベースの検証は、バックアップ操作における重要な部分です。データベースを検証するには、DBA または VALIDATE 権限が必要です。

そのため、検証には各テーブルに対する排他的なアクセスが必要です。このことを考慮すれば、データベース上に他のアクティビティがない場合に、データベースの検証を行うのが最適です。

警告
テーブルまたはデータベース全体の検証は、どの接続においてもデータベースを変更していない場合に実行してください。そうしないと、実際に破損がなくても、何らかの形でデータベースが破損したことを示す重大なエラーがレポートされます。

♦ **データベース全体の妥当性を検証するには、次の手順に従います (Sybase Central の場合)。**

1. Sybase Central の左側のウィンドウ枠で、データベースを選択します。
2. [ファイル] – [データベースの検証] を選択します。

   [データベース検証] ウィザードが表示されます。
3. ウィザードの指示に従います。

   データベースが有効かどうかを示すメッセージ・ボックスが表示されます。

ヒント
Sybase Central では、次の方法で [データベース検証] ウィザードを利用することもできます。

- データベースを右クリックし、ポップアップ・メニューから [データベースの検証] を選択します。
- データベースを選択し、[ファイル] メニューから [データベースの検証] を選択します。

♦ **データベース全体の妥当性を検証するには、次の手順に従います (SQL の場合)。**

- 次のように sa_validate ストアド・プロシージャを実行します。

  ```
  CALL sa_validate;
  ```

  プロシージャは Messages という 1 つのカラムを返します。すべてのテーブルが有効である場合、カラムには「No errors detected」と表示されます。

  詳細については、「sa_validate システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

♦ **データベース全体の妥当性を検証するには、次の手順に従います (コマンド・ラインの場合)。**

- dbvalid ユーティリティを実行します。


```
dbvalid -c "connection-string"
```

「検証ユーティリティ (dbvalid)」 756 ページを参照してください。

注意

バックアップ・コピーの妥当性を検証する場合は、どんな方法でも変更できないように、読み込み専用モードでデータベースを実行してください。バックアップ中に処理中のトランザクションがなかった場合にだけ、読み込み専用モードでデータベースを実行できます。「-r サーバ・オプション」 189 ページを参照してください。

参照

- 「データベースの妥当性の確認」 828 ページ
- 「フル・バックアップの実行」 845 ページ

単一テーブルの検証

Sybase Central または SQL 文のいずれかを使用して、単一テーブルの有効性を検証できます。テーブルを検証するには、DBA または VALIDATE 権限が必要です。

♦ テーブルの妥当性を検証するには、次の手順に従います (Sybase Central の場合)。

1. [テーブル] フォルダを開きます。
2. テーブルを右クリックし、ポップアップ・メニューから [検証] を選択します。

   データベースが有効かどうかを示すメッセージ・ボックスが表示されます。

♦ テーブルの妥当性を検証するには、次の手順に従います (SQL の場合)。

- VALIDATE TABLE 文を実行します。

```
VALIDATE TABLE table-name;
```

注意

エラーがレポートされた場合は、テーブル上のすべてのインデックスとキーをいったん削除してから再作成できます。テーブルに対する外部キーも再作成する必要があります。疑わしいインデックスがある場合は、ALTER INDEX ... REBUILD 文を実行して、破損したインデックスを再構築できます。VALIDATE TABLE がレポートするエラーに対する別の解決方法では、データベース全体をいったんアンロードし、再ロードします。データの順序付けに破損したインデックスが使用されないように、dbunload の -u オプションを使用してください。

ALTER INDEX 文の REBUILD 句の使用方法の詳細については、「ALTER INDEX 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

トランザクション・ログの検証

トランザクション・ログの検証には、対象となるトランザクション・ログがオンラインかオフラインかに関係なく、ログ変換ユーティリティ (dbtran) を使用します。ログ変換ユーティリティに

よるログ・ファイルの読み込みが正常に実行される場合、そのログは有効です。「ログ変換ユーティリティ (dbtran)」 688 ページを参照してください。

バックアップを作成し、元のトランザクション・ログを継続して使用する

ここでは、最も簡単なバックアップ方法について説明します。この作業では、トランザクション・ログを変更できません。

この種のバックアップをどのようなときに使用するかについては、「ディスク領域が十分な場合のバックアップ・スキーマ」 822 ページを参照してください。

♦ バックアップを作成し、元のトランザクション・ログを継続して使用するには、次の手順に従います (Sybase Central の場合)。

1. Sybase Central を起動します。DBA としてデータベースに接続します。
2. データベースを右クリックし、ポップアップ・メニューから [バックアップ・イメージの作成] を選択します。

   [バックアップ・イメージ作成] ウィザードが表示されます。
3. ウィザードの概要ページで、[次へ] をクリックします。
4. バックアップを作成するデータベースを選択します。
5. 次のページで、バックアップ・コピーを格納するディレクトリ名を入力し、完全なバックアップ (すべてのデータベース・ファイル) を実行するのか、インクリメンタル・バックアップ (トランザクション・ログ・ファイルのみ) を実行するのかを指定します。
6. 次のページで、[同じトランザクション・ログを継続して使用] を選択します。
7. [完了] をクリックすると、バックアップが始まります。

ヒント
Sybase Central では、次の方法を使用して [データベースのバックアップ・イメージの作成] ウィザードを利用することもできます。

- データベースを選択し、[ファイル] メニューから [バックアップ・イメージの作成] を選択します。
- データベースを右クリックし、ポップアップ・メニューから [バックアップ・イメージの作成] を選択します。

この手順では、クライアント側のバックアップを説明しています。この種類のバックアップには、他にも利用できるオプションがあります。

サーバ側のバックアップを選択したときにそのサーバが Sybase Central とは別のコンピュータで実行されている場合は、[参照] ボタンを使用してバックアップを格納するディレクトリを検索す

ることはできません。これは、[参照] ボタンを使用するとクライアント・コンピュータが検索されるのに対し、バックアップ先のディレクトリはサーバ側からの相対位置にあるためです。

♦ バックアップを作成し、元のトランザクション・ログを継続して使用するには、次の手順に従います (SQL の場合)。

· BACKUP 文を使用する場合は、次の句だけを使用します。

```
BACKUP DATABASE
DIRECTORY directory-name
[ TRANSACTION LOG ONLY ];
```

インクリメンタル・バックアップを作成する場合は、TRANSACTION LOG ONLY 句を使用します。

♦ バックアップを作成し、元のトランザクション・ログを継続して使用するには、次の手順に従います (コマンド・ラインの場合)。

· dbbackup ユーティリティを使用する場合は、次の構文を使用します。

```
dbbackup -c "connection-string" [ -t ] backup-directory
```

インクリメンタル・バックアップを作成する場合だけ -t オプションを使用します。

参照

- 「バックアップ・ユーティリティ (dbbackup)」 640 ページ
- 「BACKUP 文」 『SQL Anywhere サーバ - SQL リファレンス』

バックアップを作成し、元のトランザクション・ログを削除する

データベースがレプリケーションと関連がなく、オンライン・コンピュータ上のディスク領域に制限がある場合は、バックアップを作成するときにオンライン・トランザクション・ログを削除 (ログを「**トランケート**」) できます。この場合、データベース・ファイルで発生したメディア障害からリカバリするには、最後のフル・バックアップ以降に作成したすべてのバックアップ・コピーを使用する必要があります。

この種のバックアップをどのようなときに使用するかについては、「レプリケーションに関連しないデータベースのバックアップ・スキーマ」 823 ページを参照してください。

♦ バックアップを作成し、トランザクション・ログを削除するには、次の手順に従います (Sybase Central の場合)。

1. Sybase Central を起動します。DBA としてデータベースに接続します。
2. データベースを右クリックし、ポップアップ・メニューから [バックアップ・イメージの作成] を選択します。

   [バックアップ・イメージ作成] ウィザードが表示されます。
3. ウィザードの概要ページで、[次へ] をクリックします。

4. バックアップを作成するデータベースを選択します。
5. 次のページで、バックアップ・コピーを格納するディレクトリ名を入力し、完全なバックアップ (すべてのデータベース・ファイル) を実行するのか、インクリメンタル・バックアップ (トランザクション・ログ・ファイルのみ) を実行するのかを指定します。
6. 次のページで、[トランザクション・ログをトランケート] オプションを選択します。
7. [完了] をクリックすると、バックアップが始まります。

♦ バックアップを作成し、トランザクション・ログを削除するには、次の手順に従います (SQL の場合)。

· 次の句を使用して BACKUP 文を実行します。

```
BACKUP DATABASE
DIRECTORY backup-directory
[ TRANSACTION LOG ONLY ]
TRANSACTION LOG TRUNCATE;
```

インクリメンタル・バックアップを作成する場合だけ TRANSACTION LOG ONLY 句を使用します。

トランザクション・ログとデータベース・ファイルのバックアップ・コピーは、*backup-directory* に格納されます。パスを入力する場合、クライアント・アプリケーションではなく、データベース・サーバの作業ディレクトリとの相対パスを入力します。

♦ バックアップを作成し、トランザクション・ログを削除するには、次の手順に従います (コマンド・ラインの場合)。

· コマンド・プロンプトから次のコマンドを入力します。

```
dbbackup -c "connection-string" -x [ -t ] backup-directory
```

インクリメンタル・バックアップを作成する場合だけ -t オプションを使用します。

トランザクション・ログとデータベース・ファイルのバックアップ・コピーは、*backup-directory* に格納されます。パスを入力する場合、コマンドを実行するディレクトリとの相対パスを入力します。

注意

すべての未処理のトランザクションを終了してから、オンライン・トランザクション・ログを消去してください。未処理のトランザクションがある場合、バックアップは完了しません。「バックアップの内部メカニズム」 834 ページを参照してください。

sa_conn_info システム・プロシージャを使用すると、未処理のトランザクションがある接続を確認できます。必要に応じて、DROP CONNECTION 文を使用してユーザを切断できます。「未処理のトランザクションがある接続の特定」 853 ページを参照してください。

参照

- 「バックアップ・ユーティリティ (dbbackup)」 640 ページ
- 「BACKUP 文」 『SQL Anywhere サーバ - SQL リファレンス』

バックアップを作成し、元のトランザクション・ログの名前を変更する

通常、この一連のバックアップ・オプションは、レプリケーションに関連するデータベースに使用します。データベース・ファイルのバックアップ・コピーとトランザクション・ログを作成するだけでなく、バックアップ時のトランザクション・ログはオフライン・ログとして、名前が変更されます。新しいトランザクション・ログは、バックアップ時と同じログ名になります。

この組み合わせのバックアップ・オプションをどのようなときに使用するかについては、「レプリケーションに関連するデータベースのバックアップ・スキーマ」 824 ページを参照してください。

♦ バックアップを作成し、トランザクション・ログの名前を変更するには、次の手順に従います (Sybase Central の場合)。

1. Sybase Central を起動します。DBA としてデータベースに接続します。
2. データベースを右クリックし、ポップアップ・メニューから [バックアップ・イメージの作成] を選択します。

   [バックアップ・イメージ作成] ウィザードが表示されます。
3. ウィザードの概要ページで、[次へ] をクリックします。
4. バックアップを作成するデータベースを選択します。
5. 次のページで、バックアップ・コピーを格納するディレクトリ名を入力し、完全なバックアップ (すべてのデータベース・ファイル) を実行するのか、インクリメンタル・バックアップ (トランザクション・ログ・ファイルのみ) を実行するのかを指定します。
6. 次のページで、[ファイルの名前を変更する] オプションを選択します。
7. [完了] をクリックすると、バックアップが始まります。

♦ バックアップを作成し、トランザクション・ログの名前を変更するには、次の手順に従います (SQL の場合)。

- 次の句を使用して BACKUP 文を実行します。

  ```
  BACKUP DATABASE
  DIRECTORY backup-directory
  [ TRANSACTION LOG ONLY ]
  TRANSACTION LOG RENAME;
  ```

  インクリメンタル・バックアップを作成する場合だけ TRANSACTION LOG ONLY 句を使用します。

  トランザクション・ログとデータベース・ファイルのバックアップ・コピーは、*backup-directory* に格納されます。パスを入力する場合、クライアント・アプリケーションではなく、データベース・サーバの作業ディレクトリとの相対パスを入力します。

♦ バックアップの作成、トランザクション・ログの名前変更には、次の手順に従います (コマンド・ラインの場合)。

· コマンド・プロンプトから次のコマンドを入力します。次のように、1 行でコマンドを入力してください。

dbbackup -c "*connection-string*" -r [-t] *backup-directory*

インクリメンタル・バックアップを作成する場合は、-t オプションを使用します。

トランザクション・ログとデータベース・ファイルのバックアップ・コピーは、*backup-directory* に格納されます。パスを入力する場合、コマンドを実行するディレクトリとの相対パスを入力します。

注意

すべての未処理のトランザクションを終了してから、オンライン・トランザクション・ログの名前を変更してください。未処理のトランザクションがある場合、バックアップは完了しません。「バックアップの内部メカニズム」 834 ページを参照してください。

sa_conn_info システム・プロシージャを使用すると、未処理のトランザクションがある接続を確認できます。必要に応じて、DROP CONNECTION 文を使用してユーザを切断できます。「未処理のトランザクションがある接続の特定」 853 ページを参照してください。

参照

- 「バックアップ・ユーティリティ (dbbackup)」 640 ページ
- 「BACKUP 文」 『SQL Anywhere サーバ - SQL リファレンス』

未処理のトランザクションがある接続の特定

バージョン 8.0 以降の Adaptive Server Anywhere で作成されたデータベースのトランザクション・ログを削除したり名前を変更したりするバックアップを実行すると、完了していないトランザクションは新しいトランザクション・ログに送られます。

ただし、バージョン 7.x 以前の Adaptive Server Anywhere で作成されたデータベースのトランザクション・ログを削除したり名前を変更したりするバックアップを実行した場合には、未処理のトランザクションがあると、バックアップはそれらのトランザクションが完了するのを待ってから処理を完了します。

システム・プロシージャを使用して、未処理のトランザクションがあるユーザを判別できます。接続がそれほど多くない場合は、SQL Anywhere コンソール・ユーティリティを使用して未処理の操作がある接続を確認することもできます。

♦ 未処理のトランザクションがある接続を判別するには、次の手順に従います (SQL の場合)。

1. Interactive SQL またはその他のストアド・プロシージャを呼び出せるアプリケーションからデータベースに接続します。
2. sa_conn_info システム・プロシージャを実行します。


```
CALL sa_conn_info
```

3. **UncommitOps** カラムを検査して、未処理の操作がある接続を確認します。

   「sa_conn_info システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

♦ 未処理のトランザクションがある接続を判断するには、次の手順に従います (SQL Anywhere コンソール・ユーティリティの場合)。

1. SQL Anywhere コンソール・ユーティリティからデータベースに接続します。

   たとえば、次のコマンドでは、ユーザ ID DBA とパスワード sql を使用してデフォルト・データベースに接続します。

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

   「SQL Anywhere コンソール・ユーティリティ (dbconsole)」 718 ページを参照してください。

2. 各接続をダブルクリックし、[コミットされていない操作] のエントリを調べて、コミットされていない操作のあるユーザを確認します。必要に応じて、バックアップを終了するために、ユーザとの接続を切断できます。

バックアップ中にトランザクション・ログのバックアップ・コピーの名前を変更する

デフォルトでは、トランザクション・ログ・ファイルのバックアップ・コピーには、オンライン・ファイルと同じ名前が付けられます。バックアップを実行するたびに、バックアップ・コピーに別の名前または場所を割り当てるか、次のバックアップが終了する前にバックアップ・コピーを移動する必要があります。

トランザクション・ログのバックアップ・コピーの名前を変更して、インクリメンタル・バックアップ・コマンドを繰り返すことができます。

♦ トランザクション・ログのバックアップ・コピーの名前を変更するには、次の手順に従います (SQL の場合)。

- BACKUP 文内に MATCH キーワードを使用します。たとえば、次の文では、トランザクション・ログのインクリメンタル・バックアップをディレクトリ *c:¥backup* に作成します。トランザクション・ログのバックアップ・コピーには、*YYMMDDxx.log* 形式の名前が付きます。*YYMMDD* は日付、*xx* はカウンタであり AA から始まります。

  ```
  BACKUP DATABASE
  DIRECTORY 'c:¥¥backup'
  TRANSACTION LOG ONLY
  TRANSACTION LOG RENAME MATCH;
  ```

♦ トランザクション・ログのバックアップ・コピーの名前を変更するには、次の手順に従います (コマンド・ラインの場合)。

- dbbackup に -n オプションを指定します。たとえば、次のコマンドでは、サンプル・データベースのインクリメンタル・バックアップを作成し、トランザクション・ログのバックアップ・コピーの名前を変更します。

```
dbbackup -c "UID=DBA;PWD=sql;DBN=demo" -r -t -n c:¥backup
```

注意

トランザクション・ログのバックアップ・コピーには、*YYMMDDxx.log* 形式の名前が付きます。*YY* は年、*MM* は月、*DD* は日付です。*xx* は AA から ZZ までの英字で、1 日に何度もバックアップをする場合に値が 1 ずつ増加します。*YYMMDDxx.log* というファイル名は、順序付けではなく、識別だけを目的として使用されます。

データベースを直接テープにバックアップする

アーカイブ・バックアップでは、1 つのアーカイブ・ファイル内にデータベース・ファイルとトランザクション・ログ・ファイルのコピーを作成します。この方法を実行できるのは、サーバ側でのフル・バックアップだけです。Sybase Central でアーカイブ・バックアップを作成するときは、データベースをテープまたはディスクに直接バックアップするオプションがあります。

[データベース・バックアップ] ウィザードまたは BACKUP 文で指定するファイル名には、拡張子 *.1* が追加されます。

詳細については、「バックアップの種類」 819 ページを参照してください。

♦ アーカイブ・バックアップをテープに作成するには、次の手順に従います (Sybase Central の場合)。

1. Sybase Central を起動します。DBA としてデータベースに接続します。
2. データベースを右クリックし、ポップアップ・メニューから [データベースのバックアップ] を選択します。

   [データベース・バックアップ] ウィザードが表示されます。
3. ウィザードの概要ページで、[次へ] をクリックします。
4. テープにバックアップを作成するデータベースを選択します。
5. ウィザードの次のページでは、[デバイス上のテープ] を選択して、下にあるテキスト・ボックスにテープ・ドライブ名を入力します。
6. [完了] をクリックすると、バックアップが始まります。

♦ アーカイブ・バックアップをテープに作成するには、次の手順に従います (SQL の場合)。

- 次の句を使用して BACKUP 文を実行します。

  ```
  BACKUP DATABASE
  TO archive_root
  [ ATTENDED { ON | OFF } ]
  [ WITH COMMENT comment-string ];
  ```

  ATTENDED オプションを OFF に設定すると、テープやディスク領域が十分でない場合、バックアップは失敗します。ATTENDED を ON に設定すると、バックアップ・アーカイブ・デバイス上に領域がない場合、テープの交換などの作業をするように指示されます。

注意

BACKUP 文では、サーバ実行プログラムと同じディレクトリにあるテキスト・ファイル *backup.syb* にエントリを作成します。

アーカイブ・バックアップのリストアについては、「アーカイブ・バックアップのリストア」 860 ページを参照してください。

例

次の文では、Windows コンピュータ上で最初のテープ・ドライブにバックアップを作成します。

```
BACKUP DATABASE
TO '¥¥¥¥.¥¥tape0'
ATTENDED OFF
WITH COMMENT 'May 6 backup';
```

Windows の場合、最初のテープ・ドライブは ¥¥.¥tape0 になります。円記号は、SQL 文字列のエスケープ文字であるため、各円記号は 2 つ重ねる必要があります。

次の文では、ディスク上に *c:¥backup¥archive.1* という名前のアーカイブ・ファイルを作成します。

```
BACKUP DATABASE
TO 'c:¥¥backup¥¥archive';
```

ライブ・バックアップの作成

ライブ・バックアップを使用すると、トランザクション・ログの冗長コピーを作成できます。作成したコピーは、データベース・サーバを実行している第 1 システムが使用できなくなった場合に、第 2 システムの再起動に使用されます。ライブ・バックアップは継続的に実行され、サーバが停止した場合にのみ中止されます。システム障害が発生した場合は、バックアップされたトランザクション・ログを使って、システムをすばやく再起動できます。しかし、サーバが処理するロード量によってライブ・バックアップが遅れ、コミットされたすべてのトランザクションがバックアップされないことがあります。

通常、dbbackup ユーティリティはセカンダリ・コンピュータから実行してください。

プライマリ・コンピュータを使用できなくなった場合、セカンダリ・コンピュータを使用してデータベースを再起動できます。データベース・ファイルとトランザクション・ログは、再起動するのに必要な情報を保持しています。

ライブ・バックアップの詳細については、「コンピュータ全体に及ぶ障害からの保護」 831 ページを参照してください。

dbbackup ユーティリティに -l オプションを指定すると、トランザクション・ログのライブ・バックアップを実行できます。

♦ ライブ・バックアップを作成するには、次の手順に従います。

1. オンライン・コンピュータで障害が発生したときにデータベースを実行できるセカンダリ・コンピュータを設定します。たとえば、SQL Anywhere がセカンダリ・コンピュータ上にインストールされていることを確認します。

2. 定期的に、セカンダリ・コンピュータにフル・バックアップを実行します。

   次に例を示します。

   ```
   dbbackup -c "UID=DBA;PWD=sql;ENG=testsrv;DBN=test;LINKS=tcpip" c:¥backup
   ```

3. セカンダリ・コンピュータにトランザクション・ログのライブ・バックアップを実行します。

   ```
   dbbackup -l path¥file-name.log -c "connection-string"
   ```

4. dbbackup ユーティリティをセカンダリ・コンピュータで定期的に実行してください。

   プライマリ・コンピュータを使用できなくなった場合、セカンダリ・コンピュータを使用してデータベースを再起動できます。データベース・ファイルとトランザクション・ログは、再起動するのに必要な情報を保持しています。

データベース・ファイルのメディア障害からリカバリする

リカバリ処理で実行する必要がある手順は、バックアップ処理でインクリメンタル・バックアップのトランザクション・ログに変更を加えなかったかどうかによって異なります。バックアップの処理中にトランザクション・ログの削除または名前の変更を行った場合、複数のトランザクション・ログに加えられた変更を適用する必要があります。バックアップ処理でトランザクション・ログに変更が加えられていない場合は、リカバリのときにはオンライン・トランザクション・ログだけを使用すれば済みます。

トランザクション・ログが複数ある場合、トランザクションが複数のトランザクション・ログにまたがっている可能性があります。リカバリのときには、トランザクション・ログを正しい順序で適用する*必要があります*。適用しない場合、複数のトランザクション・ログにまたがっているトランザクションがロールバックされます。トランザクション・ログの正しい適用順序をデータベース・サーバに判断させる場合は、データベース・サーバ・オプションとして -ad を指定します。

詳細については、「複数のトランザクション・ログからのリカバリ」 862 ページを参照してください。

ここで説明したバックアップの種類の詳細については、「バックアップ・プロシージャの設計」 819 ページを参照してください。

♦ データベース・ファイルのメディア障害からリカバリするには、次の手順に従います。

1. 現在のトランザクション・ログの追加バックアップ・コピーを作成します。データベース・ファイルは失われており、最後のバックアップ以降に行われた唯一の変更の記録はトランザクション・ログにあります。
2. リカバリの処理中に使用するファイルを保存する「**リカバリ・ディレクトリ**」を作成します。
3. 最後のフル・バックアップのデータベース・ファイルをリカバリ・ディレクトリにコピーします。

4. バックアップされたトランザクション・ログに保持されているトランザクションをリカバリ・データベースに適用します。適用するには、次のいずれかの方法を使用できます。

   各トランザクション・ログをログ・ファイルごとに日付順に手動で適用するには、次の手順に従います。

   a. ログ・ファイルをリカバリ・ディレクトリにコピーします。

   b. 次のように、データベース・サーバをトランザクション・ログ適用 (-a) オプションを使用して起動し、トランザクション・ログを適用します。

      dbeng10 *database-name*.db -a *log-name*.log

      データベース・サーバは、トランザクション・ログが適用されると自動的に停止します。

   c. トランザクション・ログのバックアップをすべて適用したら、オンライン・トランザクション・ログをリカバリ・ディレクトリにコピーします。

      オンライン・トランザクション・ログのトランザクションをリカバリ・データベースに適用します。

      dbeng10 *database-name*.db -a *log-name*.log

   データベース・サーバでトランザクション・ログの正しい順序を判断して自動的に適用させるには、次の手順に従います。

   a. オンラインおよびオフラインのトランザクション・ログ・ファイルをリカバリ・ディレクトリにコピーします。

   b. データベース・サーバを -ad オプションを使用して起動し、トランザクション・ログのロケーションを指定します。データベース・サーバは、トランザクション・ログの正しい適用順序をログ・オフセットに基づいて判断します。

      dbeng10 *database-name*.db -ad *log-directory*

      データベース・サーバは、トランザクション・ログが適用されると自動的に停止します。

5. リカバリ・データベースに対して妥当性検査を実行します。

   「データベースの検証」 847 ページを参照してください。

6. リカバリ後のバックアップを作成します。

7. データベース・ファイルを運用ディレクトリに移します。

8. 運用データベースへのアクセスを許可します。

ミラーされていないトランザクション・ログのメディア障害からリカバリする

データベースが Replication Server インストール環境のプライマリ・サイト、または SQL Remote インストール環境の統合データベースにある場合、ミラーされたトランザクション・ログまたは

同様の機能を持つハードウェアを使用してください。「トランザクション・ログのメディア障害からの保護」 830 ページを参照してください。

♦ ミラーされていないトランザクション・ログのメディア障害からリカバリするには、次の手順に従います (部分リカバリの場合)。

1. ただちにデータベース・ファイルの追加バックアップ・コピーを作成します。トランザクション・ログは失われており、最後にバックアップが行われてから最新のチェックポイントまでの間に加えられた変更の唯一のレコードはデータベース・ファイルです。
2. トランザクション・ログ・ファイルを削除するか、名前を変更します。
3. -f オプションを使って、データベースを再起動します。

```
dbeng10 samples-dir¥demo.db -f
```

> **警告**
> このコマンドは、データベースが SQL Remote または Replication Server のレプリケーション・システムに関連していない場合にだけ使用してください。データベースが SQL Remote レプリケーション・システム内の統合データベースである場合は、リモート・データベースを再抽出する必要があります。

-f オプションを指定しないと、サーバはトランザクション・ログがないことを知らせるエラー・メッセージを表示します。オプションを指定すると、サーバは最新のチェックポイント時の状態にデータベースをリストアし、チェックポイントの時点でコミットされていなかったトランザクションをすべてロールバックします。その後新しいトランザクション・ログが作成されます。

samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

トランザクション・ログ・ミラーのメディア障害からリカバリする

次の手順では、ミラーされたトランザクション・ログを使用している場合におけるメディア障害からのリカバリ方法について説明します。データベースが Replication Server インストール環境のプライマリ・サイト、または SQL Remote インストール環境の統合データベースにある場合、ミラーされたトランザクション・ログまたは同様の機能を持つハードウェアを使用してください。

♦ トランザクション・ログ・ミラーのメディア障害からリカバリするには、次の手順に従います。

1. トランザクション・ログが開始された時点におけるデータベース・ファイルの追加バックアップ・コピーを作成します。
2. どちらのファイルが壊れているかを特定します。トランザクション・ログとミラーにログの変換ユーティリティを実行して、どちらがエラー・メッセージを発生するかを確認します。ログ変換ユーティリティには Sybase Central または dbtran ユーティリティからアクセスできます。

次に示すコマンド・ラインは、トランザクション・ログ *demo.log* を変換して、*demo.sql* ファイルに出力します。

```
dbtran demo.log
```

ログ変換ユーティリティでは、正常なファイルは正しく変換し、壊れたファイルにはエラーをレポートします。

3. 正しいファイルのコピーで壊れたファイルを上書きし、同一のファイルにします。
4. サーバを再起動します。

ライブ・バックアップのリカバリ

ライブ・バックアップは、運用データベースを実行するプライマリ・コンピュータとは別のコンピュータに作成されます。ライブ・バックアップからデータベースを再起動するには、セカンダリ・コンピュータに SQL Anywhere をインストールする必要があります。ライブ・バックアップの詳細については、「コンピュータ全体に及ぶ障害からの保護」 831 ページを参照してください。

♦ ライブ・バックアップを使用してデータベースを再起動するには、次の手順に従います。

1. フル・バックアップ・トランザクション・ログ・ファイルとライブ・バックアップ・トランザクション・ログを、データベース・ファイルのバックアップ・コピーに適用に使用できるディレクトリにコピーします。
2. 現在のトランザクション・ログ・ファイル名が予期されるトランザクション・ログ・ファイル名と一致する場合は、名前を変更するか削除します。
3. データベース・サーバを -ad オプションを指定して起動し、手順 1 で作成したディレクトリにあるトランザクション・ログを適用し、データベースを最新にします。

   ```
   dbeng10 samples-dir¥demo.db -ad directory-name
   ```

   samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

   データベース・サーバは、トランザクション・ログが適用されると自動的に停止します。
4. データベース・サーバを通常どおり起動して、ユーザ・アクセスを許可します。新しいアクティビティは、すべて新しいトランザクション・ログに書き込まれます。

参照

- 「コンピュータ全体に及ぶ障害からの保護」 831 ページ

アーカイブ・バックアップのリストア

アーカイブ・バックアップ (通常はテープ) を使用する場合、データのリカバリには RESTORE 文を使用します。

アーカイブ・バックアップの作成については、「データベースを直接テープにバックアップする」 855 ページを参照してください。

♦ アーカイブ・バックアップからデータベースをリストアするには、次の手順に従います (Sybase Central の場合)。

1. Sybase Central で、DBA ユーザとしてデータベースに接続します。
2. [ツール] メニューから、[SQL Anywhere 10] – [データベースのリストア] を選択します。

   [データベース・リストア] ウィザードが表示されます。
3. ウィザードの指示に従います。

♦ アーカイブ・バックアップからデータベースをリストアするには、次の手順に従います (Interactive SQL の場合)。

1. パーソナル・データベース・サーバを起動します。次のようなコマンドを使用して、restore という名前のサーバを起動します。

   ```
   dbeng10 -n restore
   ```
2. Interactive SQL を起動します。[接続] ダイアログの [ID] タブで、ユーザ ID **DBA** とパスワード **sql** を入力します。このタブにある他のすべてのフィールドはブランクのままにします。
3. [データベース] タブをクリックし、データベース名として **utility_db** と入力します。このタブにある他のすべてのフィールドはブランクのままにします。
4. [OK] をクリックして接続します。
5. アーカイブ・ルートを指定して RESTORE 文を実行します。この時点で、アーカイブされたデータベースを元のロケーション (デフォルト) にリストアするか、RENAME 句を使用して別のコンピュータに別のデバイス名を使用してリストアするかを選択できます。「RESTORE DATABASE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

例

次の文では、データベースをテープ・アーカイブからデータベース・ファイル *c:¥newdb¥newdb.db* にリストアします。

```
RESTORE DATABASE 'c:¥¥newdb¥¥newdb.db'
FROM '¥¥¥¥.¥¥tape0';
```

次の文では、データベースをファイル *c:¥backup¥archive.1* のアーカイブ・バックアップからデータベース・ファイル *c:¥newdb¥newdb.db* にリストアします。トランザクション・ログの名前と場所はデータベース内に指定されます。

```
RESTORE DATABASE 'c:¥¥newdb¥¥newdb.db'
FROM 'c:¥¥backup¥¥archive';
```

詳細については、「RESTORE DATABASE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

コミットされていない操作のリカバリ

データベース・ファイルのメディア障害からリカバリする場合、トランザクション・ログには影響ありません。リカバリを実行すると、すべてのコミットされたトランザクションがデータベースに再適用されます。状況によっては、障害が発生した時点で終了していなかったトランザクションについての情報を検索することも可能です。

[ログ・ファイル変換] ウィザードでは、Sybase Central でログ・ファイルを *.sql* ファイルに変換できます。dbtran ユーティリティを使用して、ログ・ファイルを *.sql* ファイルに変換することもできます。

♦ トランザクション・ログからコミットされていない操作をリカバリするには、次の手順に従います (Sybase Central の場合)。

1. Sybase Central で [ツール] – [SQL Anywhere 10] – [ログ・ファイルの変換] を選択します。

   [ログ・ファイル変換] ウィザードが表示されます。

2. ウィザードの指示に従います。
3. 変換されたログ (SQL コマンド・ファイル) をテキスト・エディタで編集し、必要な指示を特定します。

♦ トランザクション・ログからコミットされていない操作をリカバリするには、次の手順に従います (コマンド・ラインの場合)。

1. dbtran を実行し、トランザクション・ログを SQL コマンド・ファイルに変換します。このとき、-a オプションを指定して、コミットされていないトランザクションも含まれるようにします。たとえば、次のコマンドは、dbtran を使用してトランザクション・ログを変換します。

   ```
   dbtran -a sample.log changes.sql
   ```

2. 変換されたログ (SQL コマンド・ファイル) をテキスト・エディタで編集し、必要な指示を特定します。

   ログ変換ユーティリティの詳細については、「ログ変換ユーティリティ (dbtran)」 688 ページを参照してください。

注意

トランザクション・ログには、障害が発生する直前の変更が含まれていないこともあります。最後にコミットしたトランザクションより前にデータベースに加えられた変更は、トランザクション・ログに含まれます。

複数のトランザクション・ログからのリカバリ

SQL Anywhere では、トランザクションが複数のトランザクション・ログ・ファイルにまたがることができます。データベースがトランザクションの途中でバックアップされる場合、トランザクションは 2 つのトランザクション・ログ・ファイルにまたがることがあります。これが発生し

たら、トランザクションの最初の部分はオフライン・トランザクション・ログに、次のトランザクションの部分はオンライン・トランザクション・ログに含まれます。

データベースをリカバリする必要があり、複数のトランザクション・ログがある場合に、トランザクションが複数のトランザクション・ログにまたがっているときには*正しい順序*でトランザクション・ログ・ファイルをバックアップ・コピーに適用します。トランザクション・ログが正しい順番で適用されないと、複数のトランザクション・ログにまたがっているトランザクションの部分がロールバックされます。

トランザクション・ログを正しい順序で適用するには、次のいずれかの方法を使用します。

- -a サーバ・オプションを使用して、各ログをデータベースのバックアップ・コピーに個別に適用します。トランザクション・ログ・ユーティリティ (dblog) を使用して、トランザクション・ログ・ファイルを生成する順番を判断します。このユーティリティは、トランザクション・ログの開始時のログ・オフセットが表示されます。これによって複数のログ・ファイルを適用する順序を効果的に判断できます。

  「-a データベース・オプション」 218 ページを参照してください。

- -ad サーバ・オプションを使用して、データベースを起動し、トランザクション・ログ・ファイルのロケーションを指定します。データベース・サーバは、データベースのバックアップ・コピーに対するトランザクション・ログの正しい適用順序をログ・オフセットに基づいて判断します。

  「-ad データベース・オプション」 218 ページを参照してください。

- -ar サーバ・オプションを使用して、トランザクション・ログと同じディレクトリにある、データベースに関連付けられているログ・ファイルをいつ適用するかを、データベース・サーバに指示します。トランザクション・ログのディレクトリはデータベースから取得されます。データベース・サーバは、データベースのバックアップ・コピーに対するトランザクション・ログの正しい適用順序をログ・オフセットに基づいて判断します。

  「-ar データベース・オプション」 219 ページを参照してください。

- ログ変換ユーティリティ (dbtran) を使用して、1 つ以上のトランザクション・ログを、データベースのバックアップ・コピーに適用可能な *.sql* ファイルに変換します。「トランザクション・ログ・ユーティリティ (dblog)」 735 ページを参照してください。

-a サーバ・オプションを使用した複数のトランザクション・ログからのリカバリ

-ad サーバ・オプションを使用して、指定したディレクトリにあるすべてのトランザクション・ログ・ファイルをデータベースのバックアップ・コピーに適用することで、データベースをリカバリします。このオプションを指定すると、データベース・サーバはログを適用した後、データベースを停止します。

♦ -ad サーバ・オプションを使用して複数のトランザクション・ログからリカバリするには、次の手順に従います。

1. -ad を使用してデータベース・サーバを起動して、トランザクション・ログをデータベースのバックアップ・コピーに適用します。「-ad データベース・オプション」 218 ページを参照してください。

例

次の例では、-ad データベース・サーバ・オプションを使用して、オフライン (バックアップ) および現在のトランザクション・ログをサンプル・データベースのバックアップ・コピーに適用します。データベース・サーバは、トランザクション・ログのログ・オフセットを使用して、ログ・ファイルの正しい適用順序を判断します。

1. バックアップ・トランザクション・ログと現在のトランザクション・ログを、*c:¥backuplogs* などのディレクトリにコピーします。
2. データベース・サーバを起動し、トランザクション・ログを *backupdemo.db* という名前のデータベースのバックアップ・コピーに適用します。

   ```
   dbeng10 backupdemo.db -ad c:¥backuplogs
   ```

   データベース・サーバは、トランザクション・ログをデータベースのバックアップ・コピーに適用した後、停止します。

-a サーバ・オプションを使用した複数のトランザクション・ログからのリカバリ

-a サーバ・オプションを使用して単一のトランザクション・ログ・ファイルをデータベースのバックアップ・コピーに適用することで、データベースをリカバリします。このオプションを指定すると、データベース・サーバはログを適用した後、停止します (実行は継続しません)。複数のトランザクション・ログがある場合、ファイルを 1 つずつ正しい順序 (古いものから最新のものへ) で適用する必要があります。

♦ -a サーバ・オプションを使用して複数のトランザクション・ログからリカバリするには、次の手順に従います。

1. -a を使用してデータベース・サーバを起動して、バックアップ・トランザクション・ログをデータベースのオフライン (バックアップ) コピーに適用します。

   「-a データベース・オプション」 218 ページを参照してください。
2. データベース・サーバを起動して、現在のトランザクション・ログをデータベースのバックアップ・コピーに適用します。

例

次の例では、-a データベース・サーバ・オプションを使用して、オフライン (バックアップ) および現在のトランザクション・ログをサンプル・データベースのバックアップ・コピーに適用します。

1. データベース・サーバを起動し、*backupdemo.log* という名前のバックアップ・トランザクション・ログを *backupdemo.db* という名前のデータベースのバックアップ・コピーに適用します。

   ```
   dbeng10 backupdemo.db -a backupdemo.log
   ```

   データベース・サーバがバックアップ・トランザクション・ログをデータベースのバックアップ・コピーに適用した後、停止します。
2. データベース・サーバを起動して、*demo.log* という現在のトランザクション・ログをデータベースのバックアップ・コピーに適用します。

```
dbeng10 backupdemo.db -a demo.log
```

データベース・サーバが現在のトランザクション・ログをデータベースのバックアップ・コピーに適用した後、停止します。

dbtran ユーティリティを使用した複数のトランザクション・ログからのリカバリ

dtran により複数のトランザクション・ログを変換し、データの整合性を維持するには、-m および -n オプションの両方を指定する必要があります。-m オプションは、指定したディレクトリ内にあるログからのすべてのトランザクションを含むファイル (-n によって指定) を作成するようログ変換ユーティリティに指示します。

dbtran を使用して各ログを個別に変換する場合に、複数のトランザクション・ログ・ファイルにまたがるトランザクションがロールバックされる可能性があるため、-m を使用する必要があります。これは、dbtran がログを変換するときに、ROLLBACK 文をログの最後に追加してコミットされていないトランザクションを取り消そうとするためです。トランザクションが 2 つのログにまたがっている場合、トランザクションの COMMIT が 2 番目のログ・ファイルで発生します。最初のログ・ファイルにトランザクションの COMMIT が含まれていないために、このファイルに対する最後の操作が dbran によってロールバックされます。-m を使用してディレクトリ内のすべてのトランザクション・ログ・ファイルを変換すると、確実にすべてのトランザクションが変換されます。「トランザクション・ログ・ユーティリティ (dblog)」 735 ページを参照してください。

♦ dbtran ユーティリティを使用して複数のトランザクション・ログからリカバリするには、次の手順に従います。

1. トランザクション・ログ・ファイルを含むディレクトリに対してログ変換ユーティリティ (dbtran) を実行して、その結果生成された SQL 文を *.sql* ファイルに出力します。
2. データベースのバックアップ・コピーを開始します。
3. 手順 1 で dbtran により生成された *.sql* ファイルを、Interactive SQL からのデータベースのバックアップ・コピーに適用します。

例

次の例では、dbtran ユーティリティを使用してバックアップおよび現在のトランザクション・ログをデータベースのバックアップ・コピーに適用します。

1. *c:|backup* ディレクトリに対してログ変換ユーティリティを実行して、SQL 文を *recoverylog.sql* というファイルに出力します。

   ```
   dbtran -m "c:¥backup" -n recoverylog.sql
   ```

2. *backupdemo.db* というデータベースのバックアップ・コピーを開始します。

   ```
   dbeng10 backupdemo.db
   ```

3. *recoverylog.sql* ファイルを Interactive SQL からデータベースに適用します。

   ```
   dbisql "UID=DBA;PWD=sql;END=backupdemo" READ recoverylog.sql
   ```


トランザクション・ログの場所の変更

トランザクション・ログの場所を変更する場合、データベースは停止している必要があります。

トランザクション・ログの格納場所を選択する方法については、「データベース・ファイルで発生したメディア障害からの保護」 830 ページを参照してください。

♦ トランザクション・ログの場所を変更するには、次の手順に従います (Sybase Central の場合)。

1. [ツール] メニューから、[SQL Anywhere 10] – [ログ・ファイル設定の変更] を選択します。
2. ウィザードの指示に従います。

♦ 既存のデータベースのトランザクション・ログ・ミラーの場所を変更するには、次の手順に従います (コマンド・ラインの場合)。

1. データベース・サーバが起動していないことを確認します。
2. コマンド・プロンプトで次のコマンドを入力します。

```
dblog -t new-log-file database-file
```

「トランザクション・ログ・ユーティリティ (dblog)」 735 ページを参照してください。

トランザクション・ログ・ミラーを含むデータベースの作成

データベース作成時に、トランザクション・ログ・ミラーも保持するように指定できます。このオプションは、CREATE DATABASE 文、Sybase Central、または dbinit ユーティリティで使用できます。

トランザクション・ログ・ミラーが必要になる状況については、「トランザクション・ログのメディア障害からの保護」 830 ページを参照してください。

♦ トランザクション・ログ・ミラーを使用するデータベースを作成するには、次の手順に従います (Sybase Central の場合)。

1. Sybase Central から、[ツール] – [SQL Anywhere 10] – [データベースの作成] を選択します。
2. ウィザードの指示に従います。

♦ トランザクション・ログ・ミラーを使用するデータベースを作成するには、次の手順に従います (SQL の場合)。

- CREATE DATABASE 文に TRANSACTION LOG 句と MIRROR 句を指定して実行します。次に例を示します。

```
CREATE DATABASE 'c:¥¥mydb'
TRANSACTION LOG ON mydb.log
MIRROR 'd:¥¥mydb.mlg';
```

「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

♦ トランザクション・ログ・ミラーを使用するデータベースを作成するには、次の手順に従います (コマンド・ラインの場合)。

- -m オプションを指定して dbinit ユーティリティを実行します。たとえば、次のコマンド (1 行で入力する) は *company.db* というデータベースを初期化します。トランザクション・ログは別のデバイスに、そのミラーは更に別のデバイスに保持されます。

  ```
  dbinit -t d:¥log-dir¥company.log -m
  e:¥mirr-dir¥company.mlg c:¥db-dir¥company.db
  ```

  「初期化ユーティリティ (dbinit)」 662 ページを参照してください。

既存のデータベースでのトランザクション・ログ・ミラーの開始

トランザクション・ログ・ユーティリティを使用すると、既存のデータベースにトランザクション・ログ・ミラーを作成できます。ミラーはデータベースが実行していないときに作成します。このオプションは、Sybase Central または dblog ユーティリティで使用できます。

トランザクション・ログ・ミラーが必要になる状況については、「トランザクション・ログのメディア障害からの保護」 830 ページを参照してください。

♦ 既存のデータベースでトランザクション・ログ・ミラーを開始するには、次の手順に従います (Sybase Central の場合)。

1. [ツール] メニューから、[SQL Anywhere 10] – [ログ・ファイル設定の変更] を選択します。
2. ウィザードの指示に従います。

♦ 既存のデータベースに対してトランザクション・ログ・ミラーを開始するには、次の手順に従います (コマンド・ラインの場合)。

1. データベース・サーバが起動していないことを確認します。
2. コマンド・プロンプトで次のコマンドを入力します。

   ```
   dblog -m mirror-file database-file
   ```

   「トランザクション・ログ・ユーティリティ (dblog)」 735 ページを参照してください。

dblog ユーティリティと Sybase Central を使用して、データベースでトランザクション・ログ・ミラーを使用しないようにすることもできます。

メンテナンス・プランの作成

管理を簡素化するには、データベースに対するメンテナンス・プランを設定し、データベース・サーバで自動的に実行します。メンテナンス・プランは、データベースに対して次のタスクを 1 つ以上実行するスケジュールで構成されます。

- データベースの検証
- データベースのバックアップ
- メンテナンス・ログの管理

メンテナンス・プランは、Sybase Central から [メンテナンス・プラン作成] ウィザードを使用して作成します。メンテナンス・プランが実行されるたび、メンテナンス・レポートがデータベースに保存されます。このレポートは Sybase Central で表示できます。また、メンテナンス・プランがデータベースで実行されるたび、その後にメンテナンス・ログを電子メールで送信することもできます。

♦ メンテナンス・プランを作成するには、次の手順に従います。

1. Sybase Central でデータベースに接続します。
2. データベースの [メンテナンス・プラン] フォルダを開きます。
3. [ファイル] メニューから、[新規] – [メンテナンス・プラン] を選択します。

   [メンテナンス・プラン作成] ウィザードが表示されます。
4. ウィザードの指示に従います。

   使用できる設定の詳細については、次の項を参照してください。

   - 「スケジュールの定義」 872 ページ
   - 「VALIDATE 文」 『SQL Anywhere サーバ - SQL リファレンス』
   - 「バックアップの種類」 819 ページ
   - 「xp_startsmtp システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』

♦ メンテナンス・レポートを表示するには、次の手順に従います。

1. メンテナンス・プランの実行が完了したら、Sybase Central でデータベースに接続します。
2. [メンテナンス・プラン] フォルダを開きます。
3. 目的のメンテナンス・プランをダブルクリックします。
4. 右ウィンドウ枠にある [レポート] タブで目的のレポートをダブルクリックします。

   メンテナンス・プラン・プロパティ・シートが表示されます。[詳細] ウィンドウ枠に、メンテナンス・プランのログが表示されます。

第 21 章

スケジュールとイベントの使用によるタスクの自動化

目次

スケジュールとイベント処理の概要

データベース管理タスクの多くは、体系的に実行すると効果的です。たとえば、定期的なバックアップ手順はデータベース管理手順の重要な部分です。

データベースに「**イベント**」を追加し、イベントのスケジュールを設定することによって SQL Anywhere のルーチン・タスクを自動化できます。スケジュールに設定されている時刻になると、いつでも「**イベント・ハンドラ**」と呼ばれる一連のアクションがデータベース・サーバによって実行されます。

また、データベース管理では、ある状態が発生したときにアクションを実行することも必要です。たとえば、トランザクション・ログが格納されているディスクの空き領域が少なくなってきたときには、適切な処置を行うよう、システム管理者に電子メールで通知することが考えられます。これらのタスクも各「**システム・イベント**」に対してイベント・ハンドラを定義することによって自動化できます。

質問と回答

質問…	参照先…
スケジュールとは？	「スケジュールの概要」 872 ページ.
イベントとは?	「イベントの概要」 871 ページ
システム・イベントとは？	「システム・イベントの概要」 874 ページ
イベント・ハンドラとは？	「イベント・ハンドラの概要」 878 ページ
イベント・ハンドラのデバッグ方法は？	「イベント・ハンドラの開発」 878 ページ
データベース・サーバがスケジュールを使用してイベント・ハンドラをトリガする仕組みは？	「データベース・サーバによるスケジュールされたイベントのチェック」 880 ページ
定期バックアップをスケジュールする方法は？	「スケジュールの概要」 872 ページ.
データベース・サーバがイベント・ハンドラをトリガするのに使用できるシステム・イベントの種類は？	「システム・イベントの概要」 874 ページ 「CREATE EVENT 文」『SQL Anywhere サーバ - SQL リファレンス』
イベント・ハンドラの実行に使用される接続は？	「イベント・ハンドラの実行」 881 ページ.
イベント・ハンドラがコンテキスト情報を得る仕組みは？	「イベント・ハンドラの開発」 878 ページ 「EVENT_PARAMETER 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』

イベントの概要

データベースにイベントを追加し、イベントのスケジュールを設定することによって SQL Anywhere のルーチン・タスクを自動化できます。SQL Anywhere では、3 種類のイベントをサポートしています。

- **スケジュールされたイベント**　スケジュールに関連付けられており、指定の時間に実行されます。「スケジュールの概要」 872 ページを参照してください。
- **システム・イベント**　データベース・サーバによって追跡される特定のタイプの条件に関連付けられています。「システム・イベントの概要」 874 ページを参照してください。
- **手動イベント**　TRIGGER EVENT 文を使用して明示的に起動されます。「イベント・ハンドラのトリガ」 883 ページを参照してください。

イベント・ハンドラが実行されるたびに、エラーが発生しなかった場合、COMMIT が発生します。エラーが生成された場合は、ROLLBACK が発生します。

スケジュールの概要

アクティビティをスケジュールすると、事前に設定した時刻に一連のアクションを確実に実行できます。スケジュール情報とイベント・ハンドラはいずれもデータベース自体に格納されます。

通常は必要ありませんが、1 つの名前付きイベントに 2 つ以上のスケジュールを関連付けることによって複雑なスケジュールを定義できます。たとえば、営業時間が曜日によって変わるような販売店で、営業時間中 1 時間に 1 回イベントを発生させることができます。それぞれ別のスケジュールを持つ複数のイベントを定義して、共通のストアド・プロシージャを呼び出すことで同じような効果が得られます。

イベントをスケジュールするとき、英語の曜日をフルネーム (Monday、Tuesday など) で使用することも、省略形 (Mon、Tue など) で使用することもできます。英語以外の言語で稼働するサーバで曜日名を認識する必要がある場合は、フルネームの英語の曜日を使用してください。

次に便利なスケジュールの例を示します。

例

インクリメンタル・バックアップを毎日午前 1 時に実行します。

```
CREATE EVENT IncrementalBackup
SCHEDULE
   START TIME '1:00 AM' EVERY 24 HOURS
HANDLER
BEGIN
   BACKUP DATABASE DIRECTORY 'c:¥¥backup'
   TRANSACTION LOG ONLY
   TRANSACTION LOG RENAME MATCH
END;
```

終業時に受注の要約を作成します。

```
CREATE EVENT Summarize
SCHEDULE
   START TIME '6:00 pm'
   ON ( 'Monday', 'Tuesday', 'Wednesday', 'Thursday',
      'Friday' )
HANDLER
 BEGIN
   INSERT INTO OrderSummary
      SELECT current date,
             count( * ),
             sum( amount )
      FROM Orders
      WHERE date_ordered = current date
END;
```

参照

- 「CREATE EVENT 文」『SQL Anywhere サーバ - SQL リファレンス』

スケジュールの定義

柔軟に設定を可能にするために、スケジュール定義にはいくつかの構成要素が用意されています。

- **名前**　各スケジュール定義には名前があります。2 つ以上のスケジュールを 1 つのイベントに割り当てることができます。これは複雑なスケジュールを設計するのに便利です。
- **起動時刻**　イベントの起動時刻を定義できます。これは、イベントの実行が開始する時刻です。
- **範囲**　起動時刻の代わりとして、イベントがアクティブになる時刻の範囲を指定できます。イベントは、指定した起動時刻と終了時刻の間に発生します。頻度は指定した周期で決定します。
- **周期**　各スケジュールには周期を定義できます。イベントは、何日または何曜日ごとの何時何分何秒ごとという形で指定できる頻度でトリガされます。反復するイベントには、**EVERY** または **ON** 句が含まれています。

イベントのスケジュールは、CREATE EVENT 文に定義できるほか、[スケジュール作成] ウィザードを使用して定義することもできます。

イベントの作成時にスケジュールを追加する方法については、「CREATE EVENT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

♦ イベントのスケジュールを作成するには、次の手順に従います (Sybase Central の場合)。

1. DBA 権限のあるユーザとしてデータベースに接続します。
2. データベースの [イベント] フォルダをダブルクリックします。
3. スケジュールを作成するイベントをダブルクリックします。
4. [スケジュール] タブをクリックします。
5. [ファイル] メニューから、[新規] – [スケジュール] の順に選択します。

   [スケジュール作成] ウィザードが表示されます。
6. ウィザードの指示に従います。

システム・イベントの概要

SQL Anywhere は、いくつかのシステム・イベントを追跡します。各システム・イベントが提供するフックに一連のアクションをハングすることができます。データベース・サーバはイベントを追跡し、システム・イベントが定義された**トリガ条件**を満たしたときに (イベント・ハンドラに定義された) アクションを実行します。

トリガ条件の詳細については、「イベントのトリガ条件の定義」 875 ページを参照してください。

イベント・ハンドラを定義して、選択したシステム・イベントが発生し、定義したトリガ条件を満たしたときに実行されるようにします。このようにしておくことで、データのセキュリティが向上し、管理が容易になります。イベント・ハンドラの動作は、実行中にエラーが検出されなければコミットされ、エラーが検出された場合はロールバックされます。

使用可能なシステム・イベントは次のとおりです。

- **BackupEnd**
  BackupEnd イベント・タイプを使用すると、バックアップ終了時にアクションを実行できます。
- **DatabaseStart**
  DatabaseStart イベント・タイプを使用すると、データベース起動時にアクションを実行できます。
- **接続イベント**
  接続が確立されたとき (Connect) または接続できなかったとき (ConnectFailed)。これらのイベントはセキュリティの目的で使用できます。イベント・ハンドラに接続する代わりに、ログイン・プロシージャを使用することもできます。「login_procedure オプション [データベース]」 460 ページを参照してください。
- **Disconnect**
  Disconnect イベントを使用すると、ユーザまたはアプリケーションの切断時にアクションを実行できます。
- **ディスクの空き領域**
  データベース・ファイル (DBDiskSpace)、ログ・ファイル (LogDiskSpace)、テンポラリ・ファイル (TempDiskSpace) を格納しているデバイスの使用可能なディスク領域を追跡します。このシステム・イベントは、Windows CE では使用できません。

  ディスク領域イベントを使用すると、ディスク領域が不足したときに管理者に警告することができます。

  データベース・サーバの起動時に -fc オプションを指定して、データベース・サーバでファイル・システム・フル条件が発生した場合のコールバック関数を実装できます。「-fc サーバ・オプション」 165 ページを参照してください。
- **ファイル・サイズ**
  ファイルが指定したサイズに達したとき。これはデータベース・ファイル (GrowDB)、トランザクション・ログ (GrowLog)、テンポラリ・ファイル (GrowTemp) に使用できます。

ファイル・サイズ・イベントを使用すると、データベース上での異常なアクションを追跡したり、バルク・オペレーションをモニタすることができます。

- **GlobalAutoIncrement**
  GLOBAL AUTOINCREMENT で定義されたカラムの残りの値がこの範囲の 1 % を下回ると、GlobalAutoIncrement イベントが起動します。これは、このイベント用のパラメータとして指定された残りの値のテーブルと数字に基づいて、global_database_id オプション用の新しい値を要求するのに使用できます。このイベントにおけるテーブルの残りの値を取得するには、EVENT_PARAMETER 関数を使用し、RemainingValues パラメータと TableName パラメータを指定します。RemainingValues は、そのカラム用に生成できる残りの値の数を返します。TableName は、範囲の終わりに近づいている GLOBAL AUTOINCREMENT カラムがあるテーブルを返します。「EVENT_PARAMETER 関数 [システム]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **SQL エラー**
  RAISERROR イベント・タイプを使用すると、エラーがトリガされたときにアクションを実行できます。

- **アイドル時間**
  データベース・サーバが指定した時間アイドル状態にあったとき (ServerIdle)。このイベント・タイプを使用すると、定型の管理操作をアクセスの少ない時間に行えます。

- **データベースのミラーリング**
  プライマリ・サーバからミラー・サーバまたは監視サーバへの接続が失われると、MirrorServerDisconnect イベントが起動します。接続が失われたサーバの名前を取得するには、EVENT_PARAMETER 関数を使用し、MirrorServerName パラメータを指定します。「EVENT_PARAMETER 関数 [システム]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  サーバがデータベースの所有権を取得すると、MirrorFailover イベントが起動します。たとえば、サーバが最初に起動し、データベースを所有する必要があると判断すると、このイベントが起動します。また、直前までミラーとして動作していたサーバが、プライマリ・サーバが終了したことを特定し、監視サーバの状況を確認したうえで所有権を取得する必要があると判断するときも、このイベントが起動します。イベントは、現在ミラー・サーバとして動作しているサーバ上では起動しません。これは、データベースのコピーが依然として起動されている状態であるからです。同様に、ミラーリング・イベントを監視サーバ上で実行するように定義することはできません。ミラーリング・イベントは定義されているデータベースのコンテキスト上でのみ実行され、監視サーバはミラーされているデータベースのコピーを使用しないからです。「データベース・ミラーリングにおけるシステム・イベント」 903 ページを参照してください。

イベントのトリガ条件の定義

各イベント定義には対応するシステム・イベントがあります。また、イベント定義は 1 つまたは複数のトリガ条件を持ちます。システム・イベントに対するトリガ条件が満たされるとイベント・ハンドラがトリガされます。

トリガ条件は CREATE EVENT 文の WHERE 句に含まれていて、AND キーワードを使用して結合できます。各トリガ条件は次のフォームで定義します。

event_condition(*condition-name*)　*comparison-operator value*

condition-name 引数は、さまざまなイベント・タイプに対応できるようにあらかじめ設定されている文字列から 1 つを選択します。たとえば、**DBSize** (メガバイト単位のデータベース・ファイル・サイズ) を使用して **GrowDB** システム・イベントに適したトリガ条件を構築することができます。データベース・サーバは、条件名とイベント・タイプの対応をチェックしません。イベント・タイプのコンテキストで条件に意味があるかどうかを確認する必要があります。

例

- トランザクション・ログのサイズを 10 MB に制限します。

```
CREATE EVENT LogLimit
TYPE GrowLog
WHERE event_condition( 'LogSize' ) > 10
HANDLER
BEGIN
  IF event_parameter( 'NumActive' ) = 1 THEN
   BACKUP DATABASE
      DIRECTORY 'c:¥¥logs'
      TRANSACTION LOG ONLY
      TRANSACTION LOG RENAME MATCH;
  END IF;
END;
```

- データベース・ファイルを格納しているデバイスの空き領域が 10 % を下回ると管理者に通知しますが、ハンドラは 5 分間 (300 秒) に 2 回以上実行しません。

```
CREATE EVENT LowDBSpace
TYPE DBDiskSpace
WHERE event_condition( 'DBFreePercent' ) < 10
AND event_condition( 'Interval' ) >= 300
HANDLER
BEGIN
   CALL xp_sendmail( recipient='DBAdmin',
            subject='Low disk space',
            "message"='Database free disk space '
            || event_parameter( 'DBFreeSpace' ) );
END;
```

- データベースに侵入しようとする者を発見すると、管理者に通知します。

```
CREATE EVENT SecurityCheck
TYPE ConnectFailed
HANDLER
BEGIN
   DECLARE num_failures INT;
   DECLARE mins INT;

   INSERT INTO FailedConnections( log_time )
   VALUES ( current timestamp );

   SELECT count( * ) INTO num_failures
   FROM FailedConnections
   WHERE log_time >= DATEADD( minute, -5,
      current timestamp );

   IF( num_failures >= 3 ) THEN
      SELECT DATEDIFF( minute, last_notification,
```

```
            current timestamp ) INTO mins
        FROM Notification;


        IF( mins > 30 ) THEN
            UPDATE Notification
            SET last_notification = current timestamp;
    CALL xp_sendmail( recipient='DBAdmin',
                      subject='Security Check', "message"=
          'over 3 failed connections in last 5 minutes' )
        END IF
    END IF
END;
```

- サーバが 10 分間以上アイドル状態にあると、処理を実行します。1 時間に 2 回以上は実行しません。

```
CREATE EVENT Soak
TYPE ServerIdle
WHERE event_condition( 'IdleTime' ) >= 600
AND event_condition( 'Interval' ) >= 3600
HANDLER
BEGIN
    MESSAGE ' Insert your code here ... '
END;
```

イベント・ハンドラの概要

イベント・ハンドラは、イベントをトリガするアクションとは別の接続上で実行されます。そのため、クライアント・アプリケーションに影響することはありません。イベント・ハンドラは、イベントの作成者のパーミッションで実行されます。

イベント・ハンドラの開発

イベント・ハンドラは、スケジュールされたイベント用か、システム・イベント処理用かにかかわらず、複合文を含んでいて、多くの点でストアド・プロシージャに似ています。ループや条件付き実行などを追加することができます。また、SQL Anywhere デバッガを使ってイベント・ハンドラをデバッグすることができます。

イベント・ハンドラが実行されるたびに、エラーが発生しなかった場合、COMMIT が発生します。エラーが生成された場合は、ROLLBACK が発生します。

イベント・ハンドラのためのコンテキスト情報

イベント・ハンドラとストアド・プロシージャの違いはイベント・ハンドラが引数を取らないことです。イベントがトリガされるコンテキストについての情報は EVENT_PARAMETER 関数によって得ることができます。この関数は、イベントをトリガする接続についての情報 (接続 ID、ユーザ ID) やイベント名、実行回数などを提供します。「EVENT_PARAMETER 関数 [システム]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

イベント・ハンドラのテスト

開発中は、好きなときにイベント・ハンドラをトリガできた方が便利です。TRIGGER EVENT 文を使うと、トリガ条件やスケジュールした時刻に関係なく、明示的にイベントを実行できます。ただし、無効なイベント・ハンドラを TRIGGER EVENT によって実行することはできません。「TRIGGER EVENT 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

運用データベース上でイベント・ハンドラを開発するのはよいことではありませんが、Sybase Central から、または明示的に ALTER EVENT 文を使ってイベント・ハンドラを無効にすることができます。

コードの共有

複数のイベントを処理するアクションを 1 つにまとめておくと便利です。たとえば、データベース・ファイルまたはログ・ファイルを格納しているデバイスのディスク領域が少なくなってきたときに、通知アクションを実行することができます。これを実行するには、ストアド・プロシージャを作成し、各イベント・ハンドラの本文から呼び出します。このとき、必要なコンテキスト情報をパラメータとしてプロシージャに渡します。

イベント・ハンドラのデバッグ

イベント・ハンドラのデバッグは、ストアド・プロシージャのデバッグによく似ています。イベント・ハンドラは、イベント・リストに表示されます。

詳細と段階を追った手順については、「イベント・ハンドラのデバッグ」 884 ページを参照してください。

アクティブなイベントの制限

また、NumActive イベント・パラメータを使用して、現在アクティブになっている特定のイベント・ハンドラのインスタンスの数を判断できます。この関数は、一定時間に 1 つのイベント・ハンドラで 1 つのインスタンスだけを実行させるように制限する場合に役立ちます。

NumActive イベント・パラメータの詳細については、「EVENT_PARAMETER 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

スケジュールとイベントの内部

この項ではデータベース・サーバがスケジュールとイベント定義を処理する仕組みについて説明します。

データベース・サーバによるシステム・イベントのチェック

システム・イベントは「**イベント・タイプ**」によって分類されます。イベント・タイプは、CREATE EVENT 文の中で直接指定するか、Sybase Central を使って指定します。イベント・タイプには 2 つの種類があります。

- **アクティブ・イベント・タイプ** イベント・タイプには、データベース・サーバ自体のアクションの結果であるものがあります。こうしたアクティブなイベント・タイプには、データベース・ファイル・サイズ、さまざまなデータベース・アクションの開始時、終了時 (BackupEnd など)、RAISERROR などが含まれます。

  データベース・サーバは、アクションを実行するときに、WHERE 句に定義されたトリガ条件が満たされているかどうかをチェックし、条件が満たされていればイベント・タイプに対して定義されたイベントをトリガします。

- **ポーリング・イベント・タイプ** データベースのアクションだけではトリガされないイベント・タイプもあります。ディスクの空き領域 (DBDiskSpace など) や IdleTime タイプが含まれます。

  このタイプのイベントに対して、データベース・サーバは 30 秒ごとにポーリングします。ポーリングはデータベースの開始後、約 30 秒後から開始されます。
  IdleTime イベント・タイプの場合、データベース・サーバはサーバが 30 秒間アイドル状態にあったかどうかをチェックします。その間まったく要求が開始されず、現在アクティブな要求もなければ、秒単位のアイドル・チェック間隔時間をアイドル時間の合計に追加します。そうでない場合はアイドル時間の合計が 0 にリセットされます。したがって、IdleTime の値は、常に 30 秒の倍数になります。IdleTime がトリガ条件に指定した間隔より長くなると、IdleTime に関連付けられたイベント・ハンドラが起動します。

データベース・サーバによるスケジュールされたイベントのチェック

イベントのスケジュール時刻の計算は、データベース・サーバの起動時と、スケジュールされた各イベント・ハンドラの完了時に行われます。

次回のスケジュール時刻の計算は、スケジュール定義に指定された増分に基づいて、増分を前回の起動時刻に追加することで行われます。指定した増分よりイベント・ハンドラの実行時間が長くなり、現在の処理が終わらないうちに次回のスケジュール時刻が来てしまう場合、データベース・サーバは、現在の処理の後に次のスケジュール時刻がくるように増分します。

たとえば、実行に 65 分かかるイベント・ハンドラが 9 時 ー 5 時の間の 1 時間ごとに起動するように要求された場合、実際には 9 時、11 時、1 時と 2 時間ごとに実行されます。

次回の実行まで待機時間を設ける処理を 9 時 ー 5 時の間で実行するには、各実行の合間に WAITFOR 文を使って、指定した完了時間が経過するまでループするようにハンドラを定義できます。

データベース・サーバを断続的に実行していて、スケジュール時刻にデータベース・サーバが実行中でない場合、イベント・ハンドラが起動時に実行されることはありません。その代わりに、次のスケジュール時刻は起動時に計算されます。たとえば、毎晩 1 時にバックアップを実行するようにスケジュールしていても、終業時にはいつもデータベース・サーバを停止している場合、バックアップが実行されることはありません。

次にスケジュールされているイベント実行が 1 時間以上後の場合、データベース・サーバは時間単位で次のスケジュール時間を計算します。これにより、夏時間の開始または終了のためにシステム・クロックが調整されたときに、イベントが予定どおりに起動されます。

イベント・ハンドラの実行

イベント・ハンドラがトリガされると、一時的に内部接続が確立され、その上でイベント・ハンドラが実行されます。ハンドラが実行される接続は、ハンドラをトリガする接続とは*別*です。結果として、クライアント・アプリケーションとの対話に使用される MESSAGE ... TO CLIENT などの文は、イベント・ハンドラ内では意味を持ちません。同様に、結果セットを返す文は使用できません。

ハンドラが実行される一時的な接続は、ライセンス契約の接続制限には数えられません。login_procedure に指定したプロシージャはイベント接続では実行されません。

イベントの作成には DBA 権限が必要です。また、イベントは作成者のパーミッションで実行されます。DBA 権限を持たないでイベント・ハンドラを実行する場合、作成者のパーミッションで実行されるストアド・プロシージャのように、ハンドラ内からプロシージャを呼び出すことができます。

イベント・エラーが発生するとサーバ・コンソールに表示されます。

イベント・ハンドラとエラー
イベント・ハンドラ内のトランザクションは、実行中にエラーが検出されなかった場合はコミットされ、エラーが検出された場合はロールバックされます。
アトミックな複合文でエラーが発生し、その文に発生したエラーを処理する例外ハンドラがある場合は、その文で行われた変更は未処理のままとなります。例外ハンドラが発生したエラーを処理しない場合、または別のエラー (RESIGNAL によるエラーを含む) を発生させた場合は、そのアトミックな複合文で行われた変更は取り消されます。

イベント処理タスク

この項では、イベントを使用したタスクの自動化に関連するタスクの手順について説明します。

データベースへのイベントの追加

イベントは、Sybase Central から追加できるほか、SQL で追加することもできます。

♦ **データベースにイベントを追加するには、次の手順に従います (Sybase Central の場合)。**

1. DBA 権限のあるユーザとしてデータベースに接続します。
2. データベースの [イベント] フォルダを選択します。
3. [ファイル] メニューから [新規] – [イベント] を選択します。

   [イベント作成] ウィザードが表示されます。
4. ウィザードの指示に従います。

   ウィザードには、作成するイベントに応じて多くのオプションがあります。詳細については各タスクの中で説明しています。

♦ **データベースにイベントを追加するには、次の手順に従います (SQL の場合)。**

1. DBA 権限のあるユーザとしてデータベースに接続します。
2. CREATE EVENT 文を実行します。

   CREATE EVENT 文には、作成するイベントに応じて多くのオプションがあります。詳細については各タスクの中で説明しています。

   「CREATE EVENT 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

データベースへの手動トリガ・イベントの追加

トリガするスケジュールもシステム・イベントも設定しないでイベント・ハンドラを作成した場合、それは手動でトリガしたときのみ実行されます。

♦ **データベースに手動トリガ・イベントを追加するには、次の手順に従います (Sybase Central の場合)。**

1. DBA 権限のあるユーザとしてデータベースに接続します。
2. データベースの [イベント] フォルダを選択します。
3. [ファイル] メニューから [新規] – [イベント] を選択します。

   [イベント作成] ウィザードが表示されます。

4. イベントの名前を入力し、[次へ] をクリックします。
5. [手動] を選択し、[次へ] をクリックします。
6. [このイベントを有効にする] と [すべてのデータベースで実行] を選択してから、[次へ] をクリックします。
7. 必要に応じて、イベントを説明するコメントを入力し、[完了] をクリックしてデータベースにイベントを追加します。
8. イベント・ハンドラの SQL 文を入力し、[ファイル] – [保存] を選択します。

残りのオプションがデフォルトの値でよければ、ウィザードの早い段階で [完了] をクリックすることができます。

♦ データベースに手動トリガ・イベントを追加するには、次の手順に従います (SQL の場合)。

1. DBA 権限のあるユーザとしてデータベースに接続します。
2. スケジュールや WHERE 句なしで CREATE EVENT 文を実行します。CREATE EVENT の構文は、次のとおりです。

```
CREATE EVENT event-name
HANDLER
BEGIN
… //event handler
END
```

イベント・ハンドラを開発する場合、スケジュールやシステム・イベントを追加して、後からイベントのトリガを制御できます。これには Sybase Central または ALTER EVENT 文を使用します。

参照

- 「イベント・ハンドラのトリガ」 883 ページ
- 「ALTER EVENT 文」 『SQL Anywhere サーバ - SQL リファレンス』

イベント・ハンドラのトリガ

すべてのイベント・ハンドラはスケジュールやシステム・イベントによって起動されるほか、手動でトリガすることができます。開発中や、運用環境におけるいくつかのイベントに関しては、手動によるイベントのトリガが便利です。たとえば、毎月の売り上げレポートをスケジュールしている場合、月末でなくても売り上げレポートが必要になることがあります。

イベント・ハンドラの開発については、「イベント・ハンドラの開発」 878 ページを参照してください。

♦ イベント・ハンドラをトリガするには、次の手順に従います (Sybase Central の場合)。

1. DBA 権限のあるユーザとしてデータベースに接続します。

2. データベースの [イベント] フォルダを開きます。
3. トリガするイベントを選択し、[ファイル] – [トリガ] を選択します。

   イベントは、有効にしておかなければトリガすることはできません。イベントは、[イベント] プロパティ・シートの [一般] タブで有効にできます。

   [イベントのトリガ] ダイアログが表示されます。
4. イベント・ハンドラで必要とするパラメータをカンマで区切られたリストで次のように指定します。

   ```
   parameter=value,parameter=value
   ```

   [OK] をクリックしてイベント・ハンドラをトリガします。

♦ **イベント・ハンドラをトリガするには、次の手順に従います (SQL の場合)。**

1. DBA 権限のあるユーザとしてデータベースに接続します。
2. イベントの名前を指定して、TRIGGER EVENT 文を実行します。次に例を示します。

   ```
   TRIGGER EVENT sales_report_event;
   ```

   「TRIGGER EVENT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

イベント・ハンドラのデバッグ

いかなるソフトウェア開発でもデバッグは必要です。イベント・ハンドラは、開発プロセスでデバッグすることができます。

♦ **イベント・ハンドラをデバッグするには、次の手順に従います。**

1. Sybase Central を起動します。

   [スタート] – [プログラム] – [SQL Anywhere 10] – [Sybase Central] を選択します。
2. データベースに接続します。
3. [モード] – [デバッグ] を選択して、デバッグ・モードに変更します。

   [デバッガの詳細] ウィンドウ枠が、Sybase Central のメイン・ウィンドウの下に表示されます。
4. [フォルダ] ウィンドウ枠にある [イベント] フォルダを開きます。
5. デバッグするイベントをダブルクリックします。
6. 右ウィンドウ枠の [SQL] タブで、ブレークポイントを追加する線の左側のマージンをクリックします。赤のストップ・サインが表示され、ブレークポイントが設定されたことを示します。別の方法として、[F9] キーを押してブレークポイントを設定することもできます。

7. Interactive SQL または他のアプリケーションから、TRIGGER EVENT 文を使用してイベント・ハンドラをトリガします。
8. 設定したブレークポイントで実行が停止します。デバッガ機能を使用して実行、ローカル変数などをトレースできます。

参照

- 「イベント・ハンドラの開発」 878 ページ
- 「プロシージャ、関数、トリガ、イベントのデバッグ」『SQL Anywhere サーバ - SQL の使用法』

第 22 章

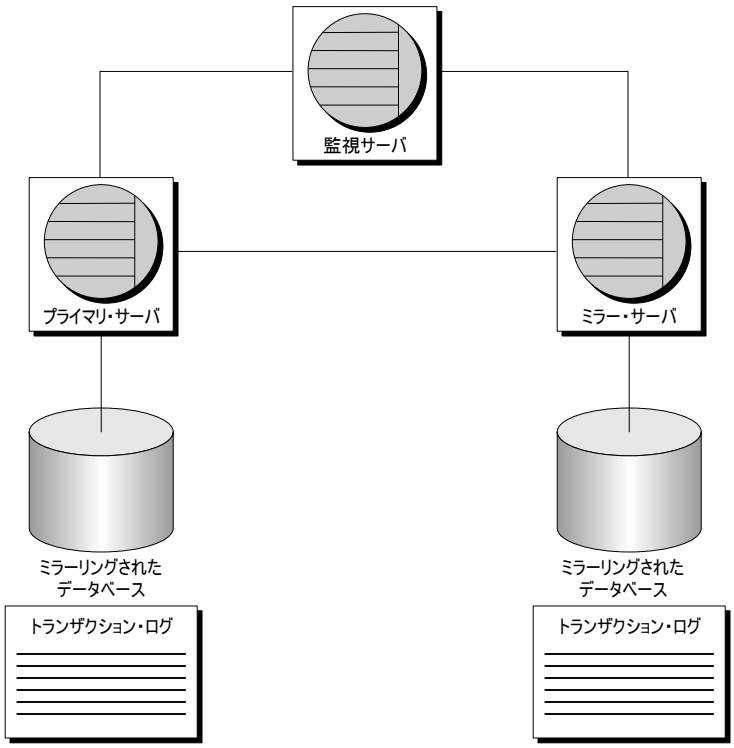
SQL Anywhere の高可用性

目次

データベース・ミラーリングの概要

「**データベース・ミラーリング**」とは、異なるコンピュータ上で実行されている 2 つまたは 3 つのデータベース・サーバが連携してデータベースやトランザクション・ログ・ファイルのコピーを保持する構成です。

「**プライマリ・サーバ**」と「**ミラー・サーバ**」は、それぞれがデータベース・ファイルとトランザクション・ログ・ファイルのコピーを保持し、「**監視サーバ**」と呼ばれる第 3 のサーバが、必要に応じて、前記 2 つのサーバのどちらにデータベースの所有権を持たせるかを決定します。監視サーバはデータベースのコピーを保持しません。この 3 種類のデータベース・サーバ (プライマリ、ミラー、監視の各サーバ) による構成は「**ミラーリング・システム**」と呼ばれ、プライマリ・サーバとミラー・サーバは併せて「**稼働サーバ**」または「**パートナー**」と呼ばれます。



クライアントがデータベースにアクセスする場合は、プライマリ・サーバに接続します。データベースに対するあらゆる変更は、プライマリ・サーバ上のトランザクション・ログ・ファイルに記録されます。変更がコミットされると、トランザクション・ログ・ページがミラー・サーバに送信され、データベースのミラー・コピーに適用されます。ミラー・サーバがミラーとして動作している間は、クライアントはそのデータベースのコピーにはアクセスできません。

プライマリ・サーバがハードウェアやソフトウェアの障害により使用できなくなった場合、ミラー・サーバが監視サーバとネゴシエートしてデータベースの所有権を取得し、プライマリ・サーバのロールを引き受けます。所有権の譲渡、すなわち「**ロールの切り替え**」を実施するには、ダウンしていない稼働サーバと監視サーバは、ロールの切り替えを行おうとする時点で、ミラーが最新で同期された状態にあることを合意する必要があります。それまで元のプライマリ・

サーバに接続されていたクライアントはすべて切断され、コミットされていないトランザクションはすべて失われます。クライアントがデータベースへのアクセスを続行するには、新しいプライマリ・サーバ上のデータベースに接続し直す必要があります。元のプライマリ・サーバは、再び使用可能になると、ミラー・サーバのロールを引き受けます。

データベース・サーバのサーバ・メッセージ・ウィンドウには、起動時に、そのサーバが引き受けているロールと起動プロセスの進捗状況を示すステータス・メッセージが表示されます。また、ミラーリング・システムを構成する他の 1 つ以上のサーバが失われたことが原因でデータベースの再起動が必要になった場合、またはロールがミラーからプライマリに変更された場合に、メッセージが表示されます。

ミラーリング・システムを構成するいずれかのサーバでアサーションが失敗すると、そのサーバはコンソールにエラーを出力して終了します。これにより、他のサーバに障害発生が通知され、適切なアクションが実行されます。

データベース・ミラーリングにハードウェアやソフトウェアに関する特別な要件はありません。また、各データベース・サーバは、地理的に離れたロケーションで実行されていてかまいません。データベース・ミラーリング・システムを構成するデータベース・サーバでは、ミラーリングされたデータベースもされていないデータベースも実行できます。また、監視サーバは、複数のデータベース・ミラーリング・システムの監視サーバになることができます。

データベース・ミラーリング・システムの各データベースのステータスの詳細は、ステータス情報ファイルに格納されます。「ステータス情報ファイル」 895 ページを参照してください。

注意
データベース・ミラーリングは、バックアップとリカバリのプランの代用ではありません。データベースには、バックアップとリカバリの手段を必ず実装してください。「データベース・ミラーリングとバックアップ」 905 ページと「バックアップとデータ・リカバリ」 811 ページを参照してください。

データベース・ミラーリングに関連する SQL Anywhere のアップグレードやデータベースの再構築の詳細については、「データベース・ミラーリング・システムでの SQL Anywhere ソフトウェアとデータベースのアップグレード」『SQL Anywhere 10 - 変更点とアップグレード』を参照してください。

クォーラム

サーバがプライマリ・サーバのロールを引き受けるには、「**クォーラム**」が必要です。これは、他のサーバの少なくともどちらか 1 つがデータベースの所有に合意していることを意味します。ミラー・サーバが使用できなくなった場合でも、プライマリ・サーバと監視サーバが接続されていれば、プライマリ・サーバは引き続きデータベースへのアクセスを提供します。プライマリ・サーバがクォーラムを失った場合、プライマリ・サーバはデータベースへのアクセスを許可できなくなります。その時点で、プライマリ・サーバは、ミラーリングされているデータベースを停止し、再起動を試行し、クォーラムを再び取得するのを待ってから、データベースをアクセス可能にします。

データベース・ミラーリング・システムが起動されると、データベース・サーバは起動プロセスを開始してクォーラムを形成し、クライアント接続を受け付けます。このプロセスにおける一般的なイベントの流れは次のとおりです。

1. 監視サーバは、サーバ 1 とサーバ 2 を待機します。
2. サーバ 1 が監視サーバまたはサーバ 2 を探します。
3. サーバ 1 が監視サーバに接続します。
4. サーバ 1 は、プライマリ・サーバになるために監視サーバとネゴシエートします。
5. 監視サーバとサーバ 1 は、サーバ 1 がプライマリ・サーバになることで合意します。
6. サーバ 1 は接続の受け付けを開始します。
7. サーバ 2 は、サーバ 1 と監視サーバを探します。
8. サーバ 2 が監視サーバとサーバ 1 に接続します。
9. サーバ 2 はクォーラムを要求します。しかし、サーバ 1 がプライマリなのでクォーラムを取得できず、サーバ 2 はサーバ 1 からのトランザクションの待機状態に入ります。
10. サーバ 1 はトランザクションをサーバ 2 に送信します。

制限

データベース・ミラーリングを使用する場合は次のような制限があります。

- **ネットワーク・データベース・サーバが必要**　ミラーリングにはサーバ間のネットワーク通信が必要なので、ネットワーク・データベース・サーバ (dbsrv10) を使用する必要があります。パーソナル・サーバは使用できません。
- **LOAD TABLE 文は許可されない**　LOAD TABLE 文は、ミラーリングされているデータベース上の永久テーブルにデータをロードしますが、ミラー・サーバではデータ・ファイルを使用できないので許可されません。LOAD TABLE 文を使用したエラーがレポートされます。ただし、一時テーブルに対しては許可されます。

  別の方法として、永久テーブルで LOAD TABLE を使用して、テンポラリ・テーブルにデータをロードし、それから INSERT ... SELECT 文を使うことができます。
- **クライアントはミラー・サーバ上のデータベースに接続できない**　サーバ停止ユーティリティ (dbstop) を使用してミラー・サーバを停止する必要が生じた場合、ユーティリティ・データベースへの接続を使用する必要があります。ユーティリティ・データベースを使用するには、データベース・サーバの起動時に -su サーバ・オプションを使用するか、ユーティリティ・データベースのパスワードを *util_db.ini* ファイルで指定します。「ユーティリティ・データベースの使用」 299 ページと「ミラーリング・システムのデータベース・サーバの停止」 902 ページを参照してください。
- **TCP/IP が必要**　ミラーリング・サーバ間で許可されるのは TCP/IP 接続だけです。
- **フェールオーバとスケジュールされたイベント**　スケジュールされたイベントのあるデータベースでフェールオーバが発生した場合、イベントの予定開始時刻の前にフェールオーバが完了していれば、スケジュールされたイベントはミラー・サーバで実行されます。完了しなかった場合、該当イベントは次にスケジュールされている時刻にミラー・サーバで実行されます。

- **トランザクション・ログの制限**　データベース・ミラーリングを使用している場合、トランザクション・ログのトランケートは実行できません。これは、トランザクションが失われる可能性があるからです。トランザクション・ログの名前は、必要に応じて何度でも変更できます。古いトランザクション・ログを削除する場合は、不要であることを確認したうえで、イベントのスケジュールを使用して削除できます。たとえば、作成されてから 1 週間が過ぎたトランザクション・ログのコピーを削除する毎日実行されるイベントを作成できます。「データベース・ミラーリングとトランザクション・ログ・ファイル」 903 ページを参照してください。
- **Web サーバはミラーリング・システムに参加できない**　データベース・ミラーリング・システムに参加している SQL Anywhere データベース・サーバは、Web サーバとしては使用できません。これは、フェールオーバが発生した場合に、データベース・サーバの IP アドレスが変更されるからです。

アプリケーション開発時の考慮事項

データベース・ミラーリングを使用する場合、アプリケーションは、ほとんどのケースでミラーリングされていないデータベースに接続している場合と同じように実行できます。ただし、データベース・ミラーリングが関係するアプリケーションを開発するうえで、考慮すべき点がいくつかあります。

- データベースに再接続できるクライアントを作成する (たとえば、フェールオーバが発生した場合、ユーザがアプリケーションをシャットダウンして再起動する必要が生じることがあります)。
- 非同期モードまたは非同期フルページ・モードで実行している場合、フェールオーバが発生してトランザクションがデータベースにコミットされない場合の対処法を決定する必要がある。
- ミラー・サーバがデータベースの所有権を引き継ぐ場合は、不完全なトランザクションをロールバックする必要があるが、トランザクションが長いほど、ロールバックに時間がかかる。フェールオーバのリカバリ速度は、クライアントの数とロールバックする必要があるトランザクションの長さに影響されます。リカバリ速度が問題になる場合は、できるだけ短いトランザクションを使用するようアプリケーションを設計してください。

データベース・ミラーリングの利点

ミラーリングには次のような利点があります。

- 監視サーバが存在する場合、プライマリからミラーへのフェールオーバは自動で実行される。同期モードで実行している場合、コミットされたトランザクションがフェールオーバ中に失われることはありません。
- ミラー・サーバにトランザクション・ログが適用済みなので、フェールオーバが非常に高速になる。ミラーがプライマリの障害を検知すると、コミットされていないトランザクションをすべてロールバックし、データベースを使用可能にします。
- 特別なハードウェア (共有ディスクなど) が不要。
- 特別なソフトウェア (クラスタリング用など) が不要。

- 特定のオペレーティング・システム・バージョン要件がない。
- サーバ同士が地理的に近い場所に設置されている必要がない。逆に、互いに距離を置くことで、火事などの災害に対する安全策が強化されます。
- ミラーリング・システムを構成するデータベース・サーバは、他のデータベースの実行にも使用できる。

監視サーバのロールの概要

監視サーバは、プライマリ・サーバ選定におけるサーバ間の競合を解消します。監視サーバがないと、サーバ A の起動時にサーバ B が使用できなかった場合に、サーバ A は自身のデータベース・ファイルのコピーが最新の状態なのかを判断できません。最新ではないファイルを使用してデータベースを起動すると、もう一方のデータベースのコピーに適用されてコミットされたトランザクションが失われます。また、2 つの稼働サーバが接続を再確立した後では、もう一方のデータベースのコピーをミラーリングに使用できなくなります。

監視サーバは、起動時の競合を解消するほか、2 つのサーバが、サーバ間の通信リンクが壊れたままで引き続き実行されている場合についても対応します。監視サーバがなかった場合、どちらのサーバもデータベースの所有権を取得しようとします。この場合も、トランザクションが失われ、それぞれ互換性のないデータベースになってしまいます。監視サーバがあれば、プライマリ・サーバはデータベースの所有権が存続することを証明して、クライアントにアクセスを提供し続けることができます。プライマリ・サーバがミラーと監視サーバのどちらとも通信できなくなった場合は、シャットダウンして、どちらかに接続可能になるまで待機する必要があります。

監視サーバは、複数のミラーリング・システムの監視サーバとして機能できます。また、他のデータベースのデータベース・サーバとしても機能できます。

データベース・ミラーリングで使用するモードの選択

ミラーリングには次の 3 種類の実行モードがあります。

- 同期
- 非同期
- 非同期フルページ

デフォルトは同期モードです。各モードは、トランザクションがミラー・サーバで記録されるタイミングと方法を制御します。設定には、-xp サーバ・オプションを使用します。

データベース・ミラーリング・システムで使用する同期実行モードを選択する場合は、フェールオーバの発生時にリカバリ速度とデータの状態とでどちらを優先するかを決定する必要があります。

同期モード

同期モードでは、コミットされたトランザクションはミラー・サーバに記録されることが保証されます。プライマリ・サーバで障害が発生した場合、ミラー・サーバが引き継いだときに、コミットされたトランザクションが失われることはありません。このモードでは、プライマリ・

サーバはトランザクションがコミットされるとトランザクション・ログ・ページをミラーに送信します。ミラー・サーバは、送信されたページをトランザクション・ログのコピーに書き込むと、転送の受信確認を行います。プライマリ・サーバは、受信確認を受信してからアプリケーションに応答します。

同期モードを使用することで「**トランザクションの安全性**」が保証されます。これは、稼働サーバが同期された状態になり、プライマリは、ミラーに送信した変更の受信確認を待ってから、処理を続行するからです。

非同期モード

非同期モードでは、コミットされたトランザクションがミラー・サーバに記録されることは保証されません。このモードでは、プライマリ・サーバはトランザクションがコミットされるとトランザクション・ログ・ページをミラーに送信します。この時点で、プライマリはミラーからの受信確認を待たずに、COMMIT の完了をアプリケーションに応答します。このため、プライマリ・サーバで障害が発生した場合、ミラー・サーバが引き継いだときにコミット済みのトランザクションが一部失われる可能性があります。

非同期フルページ・モード

非同期フルページ (または page) モードでは、ページは COMMIT の実行時にではなく、ページが満杯になったときに送信されます。これにより、2 つのデータベース・サーバ間のトラフィック量が減り、プライマリ・サーバのパフォーマンスが向上します。現在のログ・ページは、pagetimeout パラメータに指定した秒数を過ぎてもミラーに送信されなかった場合に、ページが満杯でなくても送信されます。デフォルトの値は 5 秒です。このモードを使用すると、プライマリ・サーバがダウンし、ミラー・サーバがデータベースの所有権を引き継ぐ場合に、コミットされたトランザクションが失われる可能性がある状態に置かれている時間を制限できます。非同期フルページ・モードは非同期操作の一種であるため、プライマリ・サーバはミラーからの受信確認を待機しません。

非同期モードおよび非同期フルページ・モードは、同期モードより高速ですが、上記の理由により信頼性は低下します。非同期モードおよび非同期フルページ・モードでは、プライマリ・サーバに適用されたコミット済みトランザクションがミラー・サーバにすべて適用されているとは限らないので、プライマリ・サーバからミラー・サーバへのフェールオーバは自動ではありません。このため、非同期のどちらかのモードを使用する場合は、ミラー・サーバはデフォルトで、プライマリに障害が発生してもデータベースの所有権を取得できません。この場合に (トランザクションが失われる可能性があったとしても) 自動フェールオーバを使用したい場合は、-xp サーバ・オプションを使用して autofailover オプションを yes に設定します。それ以外の場合、障害が発生したサーバは、再起動後にトランザクションが失われたかどうかを確認します。トランザクションが失われていた場合、メッセージをサーバ・コンソールに出力して、データベースをシャットダウンします。現在のデータベースとトランザクション・ログは、ミラーリングを続行する前にバックアップを使用して置き換える必要があります。

非同期モードまたは非同期フルページ・モードで障害が発生した後にサーバを再起動する方法については、「プライマリ・サーバ障害からのリカバリ」 902 ページを参照してください。

注意
非同期モードまたは非同期フルページ・モードを使用している場合は、-xp autofailover オプションを yes に設定することをおすすめします。それによって、プライマリ・サーバで障害が発生した場合、ミラー・サーバが自動的にプライマリ・サーバとなります。

synchronize_mirror_on_commit オプションでは、非同期モードまたは非同期フルページ・モードにおいて、データベースの変更がミラー・サーバへ送信されたことがどの時点で保証されるかについて制御できます。このオプションを On に設定すると、COMMIT が発生するたびに、トランザクション・ログに記録されたすべての変更がミラー・サーバへ送信され、それの変更がミラー・サーバによって受信されると、ミラー・サーバからプライマリ・サーバへ受信確認が送信されます。このオプションは、SET TEMPORARY OPTION を使用して特定のトランザクションに対して設定できます。また、ログイン・プロシージャで APPINFO 文字列を調べ、このオプションを特定のアプリケーションだけに設定すると、効果的な場合もあります。

SQL Anywhere では、データベース・ミラーリング・システムでフェールオーバが発生したときに、使用しているモードに関係なく起動するシステム・イベントをサポートしています。このようなイベントを使用することで、フェールオーバの発生時に管理者に通知することなどができます。「データベース・ミラーリングにおけるシステム・イベント」 903 ページを参照してください。

参照

- 「synchronize_mirror_on_commit オプション [データベース]」 502 ページ
- 「-xp データベース・オプション」 227 ページ
- 「SET OPTION 文」『SQL Anywhere サーバ - SQL リファレンス』

同期ステータス

同期モードを使用しているミラーリング・システムは、「同期中」または「同期」のどちらかのステータスになります。

稼働サーバのどれかが起動し、ミラーとして動作することになった場合、まず取得していないログ・ページをプライマリ・サーバに要求します。この処理には、プライマリ・サーバ上で現在アクティブなログ以外のログ・ファイルからページをコピーすることも含まれます。ミラーは、ページを受信しては、そこに含まれている変更を自身のデータベースのコピーに適用します。ミラーがプライマリからすべてのページを受信すると、プライマリとミラーは同期した状態になります。この時点以降、プライマリ上でコミットされたあらゆる変更はミラーに送信され、ミラーによって受信確認される必要が生じます。

非同期モードと非同期フルページ・モードでは、ミラーは上記と同様にログ・ページを要求します。ただし、2 つのサーバが同期した状態になることはありません。ミラーがプライマリから取得できるすべてのログ・ページを要求すると、プライマリは、ページが更新されたらすべてミラーに送信するよう通知されます。

ステータス情報ファイル

ミラーリング・システムの各サーバはステータス情報ファイルを管理し、ミラーリング・システムのステータスについて各サーバの視点から記録を残します。

ステータス情報ファイルは、起動時に、サーバが引き受けるロールの決定に使用されます。サーバのローカル・ステータスは、データベース・ミラーリング・システム内の他のサーバのステータスと比較されます。ミラーリング・システムの各サーバのステータス情報ファイルは、-xf オプションを使用して必ず指定する必要があります。「-xf サーバ・オプション」 207 ページを参照してください。

ステータス情報ファイルに含まれている情報は次のとおりです。

フィールド	説明
Owner	どのデータベース・サーバがプライマリ・サーバかを示す。
State	サーバがログ・ページの受信中なのか、または最新状態なのかを示す同期ステータス (「同期中」または「同期」のどちらか) を示す。「同期ステータス」 894 ページを参照してください。
Mode	同期実行モード (同期、非同期、非同期フルページのいずれか) を示す。「データベース・ミラーリングで使用するモードの選択」 892 ページを参照してください。
Sequence	データベース・ミラーリング・システムでフェールオーバが発生した回数を示す。このシーケンス番号はロールの切り替えが発生するたびに 1 ずつ増分されます。これは、サーバの視点でミラーリング・システムの状態が最新かどうかを判断するのに役立ちます。「データベース・ミラーリングの概要」 888 ページを参照してください。

ステータス情報ファイルの内容の例を次に示します。

```
[demo]
Owner=server2
State=synchronizing
Mode=asynchronous
Sequence=35
```

ステータス情報ファイルがない場合は、自動的に作成されます。ステータス情報ファイルの変更は、データベース・サーバのみが行います。

チュートリアル：データベース・ミラーリングの使用

このチュートリアルでは、データベース・ミラーリング・システムの設定方法とフェールオーバが発生したときの動作を説明します。このチュートリアルでは、すべてのデータベース・サーバが同じコンピュータ上で実行されていることを想定しています。ただし、実際のミラーリング・システムでは、データベース・サーバを異なるコンピュータ上で実行するのが一般的です。

♦ データベース・ミラーリング・システムでのフェールオーバをシミュレートするには、次の手順に従います。

1. ディレクトリ *c:¥server1*、*c:¥server2*、*c:¥arbiter* を作成します。
2. *samples-dir¥demo.db* に置かれているサンプル・データベースのコピーを作成し、*c:¥server1* に追加します。

   samples-dir の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。
3. 次のコマンドを実行して、*c:¥server1* にあるデータベース用のトランザクション・ログを作成します。

   ```
   dbping -d -c UID=DBA;PWD=sql;DBF=c:¥server1¥demo.db
   ```
4. *c:¥server1* にあるデータベース・ファイルとトランザクション・ログのコピーを作成し、*c:¥server2* に追加します。
5. コマンド・プロンプトで次のコマンドを実行して、監視サーバを起動します。

   ```
   dbsrv10 -x tcpip(PORT=2639) -su sql -n arbiter -xa auth=abc;DBN=demo -xf c:¥arbiter
   ¥arbiterstate.txt
   ```

   このコマンド・ラインは、以下に示す dbsrv10 のオプションを指定します。

   - **-x**　データベース・サーバに対し、TCP/IP 通信にポート 2639 を使用するよう指示します。他のサーバも TCP/IP を使用しますが、通信には別のポートを使用します。
   - **-su**　ユーティリティ・データベースのパスワードを指定します。
   - **-n**　データベース・サーバに arbiter という名前を付けます。
   - **-xa**　ミラーリング対象のデータベースの名前と、監視サーバに対する認証文字列 (ここでは abc) を指定します。この認証文字列を、データベース・ミラーリング・システムのすべてのサーバ (監視、プライマリ、ミラー) で使用する必要があります。
   - **-xf**　arbiter のステータス情報ファイルのロケーションを指定します。
6. コマンド・プロンプトで次のコマンドを実行して、server1 を起動します。コマンドは 1 行で入力してください。

   ```
   dbsrv10 -n server1 -x tcpip(PORT=2638) -xf c:¥server1¥server1state.txt -su sql c:¥server1¥demo.db
   -sn mirrordemo
   -xp partner=(ENG=server2;LINKS=tcpip(PORT=2637;TIMEOUT=1));auth=abc;
   arbiter=(ENG=arbiter;LINKS=tcpip(PORT=2639;TIMEOUT=1));mode=sync
   ```

   このコマンド・ラインは、以下に示す dbsrv10 のオプションを指定します。

   - **-n**　データベース・サーバに server1 という名前を付けます。
   - **-x**　データベース・サーバの実行で使用するポート番号を指定します。
   - **-xf**　server1 のステータス情報ファイルのロケーションを指定します。
   - **-su**　ユーティリティ・データベースのパスワードを指定します。

- **-sn**　データベース・サーバの代替名を指定します。プライマリ・サーバとミラー・サーバは同じ名前にしておき、どちらがプライマリでどちらがミラーなのかをクライアントが事前に知らなくても接続できるようにします。
- **-xp**　起動中のサーバがパートナと監視サーバに接続できるよう情報を提供します。

7. コマンド・プロンプトで次のコマンドを実行して、server2 を起動します。コマンドは 1 行で入力してください。

```
dbsrv10 -n server2 -x tcpip(PORT=2637) -xf c:¥server2¥server2state.txt -su sql c:¥server2¥demo.db
-sn mirrordemo
-xp partner=(ENG=server1;LINKS=tcpip(PORT=2638;TIMEOUT=1));auth=abc;
arbiter=(ENG=arbiter;LINKS=tcpip(PORT=2639;TIMEOUT=1));mode=sync
```

このコマンド・ラインは、以下に示す dbsrv10 のオプションを指定します。

- **-n**　データベース・サーバに server2 という名前を付けます。
- **-x**　データベース・サーバの実行で使用するポート番号を指定します。
- **-xf**　server2 のステータス情報ファイルのロケーションを指定します。
- **-su**　ユーティリティ・データベースのパスワードを指定します。
- **-sn**　データベース・サーバの代替名を指定します。プライマリ・サーバとミラー・サーバは同じ名前にしておき、どちらがプライマリでどちらがミラーなのかをクライアントが事前に知らなくても接続できるようにします。
- **-xp**　起動中のサーバがパートナと監視サーバに接続できるよう情報を提供します。

8. 次のコマンドを実行して、Interactive SQL を起動し、プライマリ・サーバに接続します。

```
dbisql -c "UID=DBA;PWD=sql;ENG=mirrordemo;LINKS=tcpip"
```

9. 次の文を実行して、サンプル・データを SQL Anywhere サンプル・データベースに追加します。

```
CREATE TABLE test (col1 INTEGER, col2 CHAR(32));
INSERT INTO test VALUES(1, ‘Hello from server1’);
COMMIT;
```

10. 次の文を実行して、接続したデータベース・サーバを確認します。

```
SELECT PROPERTY( 'ServerName' );
```

プライマリ・サーバの名前が出力されます。

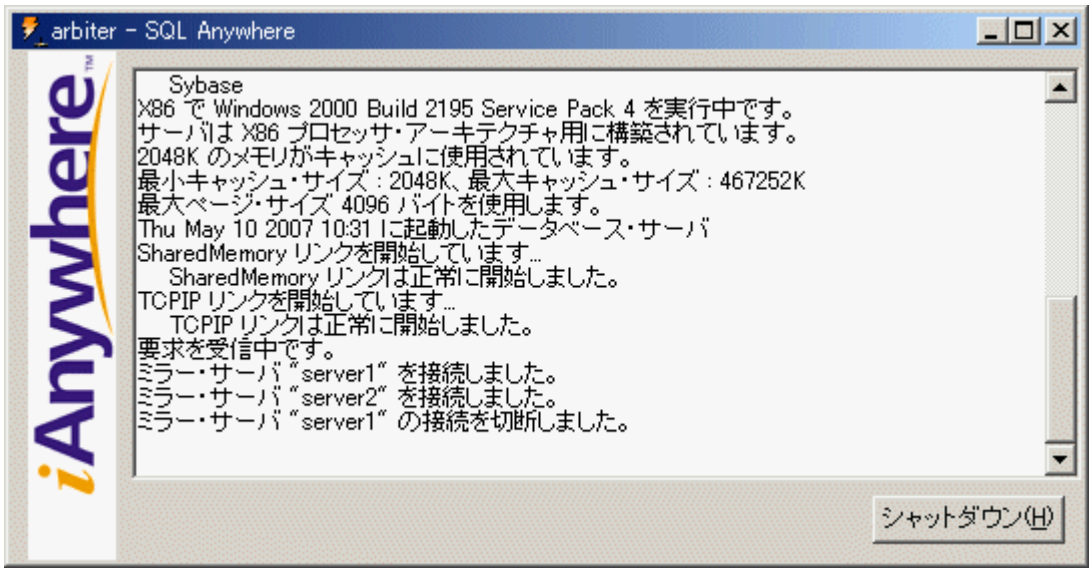
11. フェールオーバを開始します。これを行うには、次のいずれかの方法で、手順 10 で開始したプライマリ・サーバを停止します。

- サーバ・メッセージ・ウィンドウで [シャットダウン] をクリックします。
- Windows のタスク マネージャを使用してタスクを終了します。
- 次のコマンドを発行します。

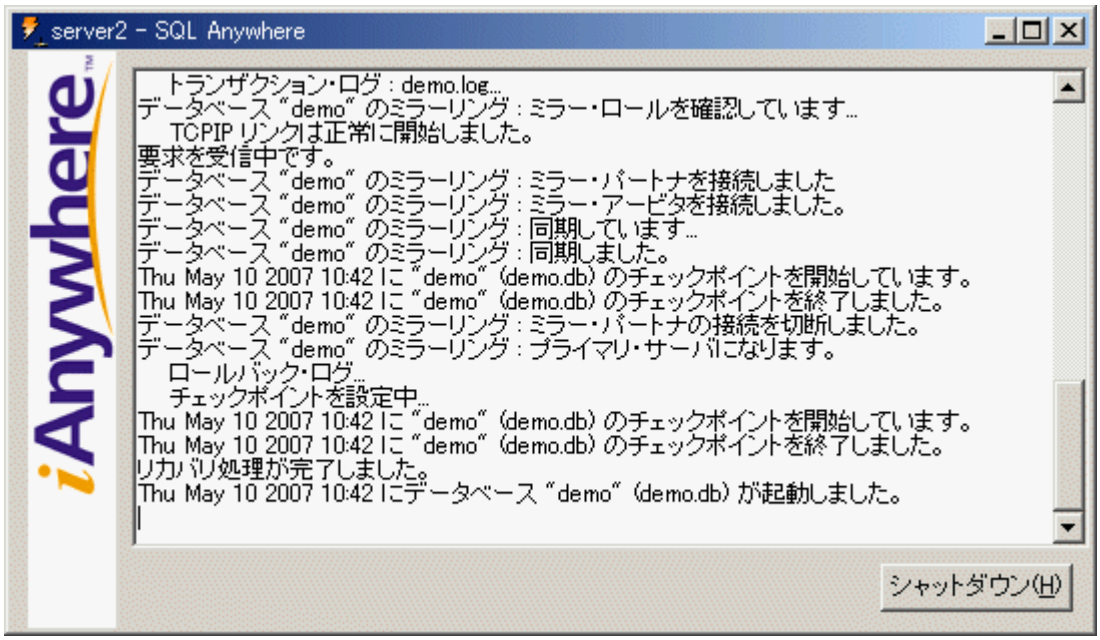
```
dbstop -y -c "UID=DBA;PWD=sql;ENG=mirrordemo"
```


データベース・サーバの 1 つの接続が有効であるという警告メッセージが表示された場合は、[はい] をクリックしてシャットダウンします。

arbiter のサーバ・メッセージ・ウィンドウに、プライマリ・サーバが切断されたことを示すメッセージが表示されます。



server2 のサーバ・メッセージ・ウィンドウには、server2 が新しいプライマリ・サーバであることを示すメッセージが表示されます。



12. Interactive SQL を閉じます。エラー・メッセージを受信した場合は [OK] をクリックします。
13. 次のコマンドを実行して、Interactive SQL を再起動します。

    ```
    dbisql -c "UID=DBA;PWD=sql;ENG=mirrordemo;LINKS=tcpip"
    ```

14. 次の文を実行して、ミラー・サーバに接続されているかどうかを確認します。

    ```
    SELECT PROPERTY ( 'ServerName' );
    ```

15. 次の文を実行して、すべてのトランザクションがミラー・サーバにミラーリングされたかどうかを確認します。

    ```
    SELECT * FROM test;
    ```

16. Interactive SQL から切断し、arbiter、server1、server2 のサーバ・メッセージ・ウィンドウで [シャットダウン] をクリックします。

データベース・ミラーリングの設定

次の手順では、ミラーリング・システムの構築対象となるデータベースを実行しているデータベース・サーバがすでに 1 つ存在していることを想定しています。

ミラーリング・システムに参加するデータベース・サーバを起動する場合は、-su オプションを使用して、ユーティリティ・データベースのパスワードを指定することをおすすめします。これにより、ユーティリティ・データベースを使用してサーバをシャットダウンしたり、必要に応じてミラー・サーバを強制的にプライマリ・サーバにしたりできるようになります。「-su サーバ・オプション」 196 ページを参照してください。

データベース・ミラーリングに関連する SQL Anywhere のアップグレードやデータベースの再構築の詳細については、「データベース・ミラーリング・システムでの SQL Anywhere ソフトウェアとデータベースのアップグレード」 『SQL Anywhere 10 - 変更点とアップグレード』を参照してください。

♦ ミラーリング・システムを設定するには、次の手順に従います。

1. 2 つ目のサーバ上に、データベースと現在のトランザクション・ログのコピーを作成します。

   既存のデータベース・サーバが停止している場合は、ファイルのコピーを作成できます。そうでない場合は、BACKUP DATABASE 文またはバックアップ・ユーティリティ (dbbackup) を使用します。「BACKUP 文」 『SQL Anywhere サーバ - SQL リファレンス』と「バックアップ・ユーティリティ (dbbackup)」 640 ページを参照してください。

2. 実行中のデータベース・サーバを停止し、ミラーリング・オプションを含むようにコマンド・ライン設定を変更して、サーバを再起動します。

   次に例を示します。

   ```
   dbsrv10 -n server1 -x tcpip(PORT=2638) -xf c:¥server1¥server1state.txt -su sql c:¥server1
   ¥mirrordemo.db -sn mirrordemo
   -xp partner=(ENG=server2;LINKS=tcpip(PORT=2637;TIMEOUT=1));auth=abc;
   arbiter=(ENG=arbiter;LINKS=tcpip(PORT=2639;TIMEOUT=1));mode=page;autofailover=YES
   ```


3. もう一方の稼働サーバを起動します。

   次に例を示します。

   ```
   dbsrv10 -n server2 -x tcpip(port=2637) -xf c:¥server2¥server1state.txt -su sql c:¥server2
   ¥mirrordemo.db -sn mirrordemo
   -xp partner=(ENG=server1;LINKS=tcpip(PORT=2638;TIMEOUT=1));auth=abc;
   arbiter=(ENG=arbiter;LINKS=tcpip(PORT=2639;TIMEOUT=1));mode=page;autofailover=YES
   ```

4. 監視サーバを起動します。

   次に例を示します。

   ```
   dbsrv10 -x tcpip -n arbiter
   -xa AUTH=abc;DBN=mirrordemo -xf arbiterstate.txt
   -su sql
   ```

   これで、クライアントはミラーリングされたデータベースに接続できるようになりました。

ミラーリングされたデータベース・サーバへの接続

ミラーリングされたデータベースに接続する場合、クライアントは、プライマリ・サーバとミラー・サーバの起動に使用されたコマンドの -sn オプションに指定されたサーバ名を使用する必要があります。上記の例 (データベース・サーバは -sn mirrordemo で起動) では、クライアントは接続パラメータ ENG=mirrordemo を接続文字列に指定しています。

```
…UID=user12;PWD=x92H4pY;ENG=mirrordemo;LINKS=tcpip…
```

プライマリ・サーバとミラー・サーバが異なるサブネット上で実行されている場合は、クライアントがプライマリ・サーバに接続するために使用する IP アドレスの範囲を指定する必要があります。次に例を示します。

```
…UID=user12;PWD=x92H4pY;ENG=mirrordemo;LINKS=tcpip(HOST=ip1,ip2…)…
```

クライアントがプライマリ・サーバへの接続を試行し続ける時間を制御するために、RetryConnectionTimeout 接続パラメータを指定することもできます。「RetryConnectionTimeout 接続パラメータ [RetryConnTO]」 261 ページを参照してください。

クライアントの接続先となるサーバが見つからない場合は、以下を試してください。

1. プライマリ・サーバとミラー・サーバを実行しているコンピュータのホスト名を指定します。たとえば、MirrorServ1 および MirrorServ2 という名前のコンピュータ上で実行されている場合、クライアントの接続文字列で LINKS=tcpip(HOST=MirrorServ1,MirrorServ2) と指定できます。
2. LDAP にサーバを登録します。「LDAP サーバを使用した接続」 126 ページを参照してください。
3. SQL Anywhere Broadcast Repeater ユーティリティ (dbns10) を使用してサーバを探します。このユーティリティは、あるサブネット上でのブロードキャストと応答を受信し、それを別のサブネットに再ブロードキャストします。「SQL Anywhere Broadcast Repeater ユーティリティ (dbns10)」 715 ページを参照してください。

優先データベース・サーバの指定

データベース・ミラーリング・システムでは、2 つの稼働サーバのうちいずれかを優先サーバとして指定できます。すべてのデータベース・サーバが実行されている場合、優先サーバがプライマリ・サーバとなり、データベースの所有権を持ちます。優先サーバとしてマークされているサーバが使用できなくなると、ミラー・サーバとして稼働していたサーバがプライマリ・サーバとなります。優先サーバが再開すると、優先サーバは、まだ所有していないすべてのトランザクション・ログ・エントリを現在のプライマリ・サーバから取得します。その後、現在のプライマリ・サーバに対して、データベースの所有権の放棄を確認します。優先サーバと現在のプライマリ・サーバはロールを変更し、優先サーバがプライマリ・サーバになり、もう一方のサーバがミラー・サーバになります。データベースの所有権が変更されると、優先サーバではないサーバにあるデータベースへの接続は失われます。

データベース・サーバの起動時、-xp データベース・オプションに "preferred=YES" を追加して、優先サーバを指定します。次に例を示します。

```
dbsrv10 -n server1 mydata.db -sn mydata
-xp partner=(ENG=server2;LINKS=tcpip(TIMEOUT=1));
AUTH=abc;arbiter=(ENG=arbsrv;LINKS=tcpip(TIMEOUT=1));preferred=YES
```

参照

- 「-xp データベース・オプション」 227 ページ
- 「プライマリ・サーバのフェールオーバの起動」 901 ページ
- 「データベース・ミラーリングで使用するモードの選択」 892 ページ

データベース・サーバの強制プライマリ・サーバ化

プライマリ・サーバを強制的にシャットダウンする必要が生じた場合 (実行用のコンピュータを置き換える場合など)、ALTER DATABASE 文を使用して、ミラー・サーバを強制的にプライマリ・サーバにできます。この機能を使用するには、データベース・サーバ上のユーティリティ・データベースに接続できる必要があります。ユーティリティ・データベースに接続するには、コマンドに -su オプションを指定してミラー・サーバを起動するか、*util_db.ini* ファイルにパスワードを指定します。次のコマンドにより、データベース *mymirroreddb.db* のミラー・サーバが強制的にプライマリ・サーバになります。

```
ALTER DATABASE mymirroreddb FORCE START;
```

詳細については、「ALTER DATABASE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

現在ミラー・サーバとして動作しているデータベース・サーバに、データベースの所有権の取得を強制します。この文は、プライマリ・サーバ上のデータベースに接続しているときに実行する必要があります。プロシージャまたはイベントから実行できます。

プライマリ・サーバのフェールオーバの起動

次の文を実行すると、プライマリ・サーバからミラー・サーバへのデータベース・ミラーリングのフェールオーバを起動できます。


```
ALTER DATABASE SET PARTNER FAILOVER
```

この文は、優先サーバを指定する代わりに使用できます。また、特定のデータベース・サーバに対してデータベースの所有権が指定されているときに制御するロジックで使用することができます。たとえば、パートナ・サーバ (PartnerState データベース・プロパティの値で決まります) の可用性、またはデータベースに対する接続数 (ConnCount データベース・プロパティの値で決まります) に基づいて、フェールオーバを起動できます。

この文を実行すると、データベースに対する既存の接続は、文を実行した接続も含めて、閉じられます。結果として、文がプロシージャまたはイベントに含まれている場合、後続する他の文は実行されない可能性があります。この文の実行に必要なパーミッションは、-gk サーバ・オプションで制御します。

参照

- 「優先データベース・サーバの指定」 901 ページ
- 「ALTER DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「-gk サーバ・オプション」 170 ページ
- ConnCount プロパティと PartnerState プロパティ：「データベース・レベルのプロパティ」 550 ページ

ミラーリング・システムのデータベース・サーバの停止

プライマリ・サーバ、ミラー・サーバ、または監視サーバを停止する必要が生じる場合があります。停止するには、データベース停止ユーティリティ (dbstop) を使用します。サーバを停止するにはユーティリティ・データベースへの接続を使用する必要があります。このため、データベース・サーバの起動時には -su サーバ・オプションを含めることをおすすめします。「ユーティリティ・データベースの使用」 299 ページを参照してください。

- **プライマリ・サーバ、ミラー・サーバ、監視サーバを停止するには、次の手順に従います。**

1. dbstop コマンドを発行して、データベース・サーバを停止します。

   たとえば、次のコマンドを実行するとデータベース・サーバ myarbiter が停止します。

   ```
   dbstop -c "UID=DBA;PWD=sql;DBN=utility_db;LINKS=tcpip" myarbiter
   ```

プライマリ・サーバ障害からのリカバリ

プライマリ・サーバ障害からのリカバリ手順は、データベース・ミラーリング・システムで使用している同期実行モードによって異なります。

同期モードで実行している場合は、プライマリ・サーバに存在するすべてのトランザクションがミラー・サーバでもコミットされていることが保証されます。ミラー・サーバは、ユーザによる介入なしに、新しいプライマリ・サーバになることができます。

非同期モードおよび非同期フルページ・モードでは、プライマリ・サーバに適用されたコミット済みトランザクションがミラー・サーバにすべて適用されているとは限らないので、プライマリ・サーバからミラー・サーバへのフェールオーバは自動ではありません。このため、自動フェー

ルオーバの実行が指定されていないかぎり、非同期のどちらかのモードを使用する場合は、ミラー・サーバはデフォルトで、プライマリに障害が発生してもデータベースの所有権を取得できません。障害が発生したサーバは、再起動後にトランザクションが失われたかどうかを確認します。トランザクションが失われていた場合、メッセージをサーバ・コンソールに出力して、データベースをシャットダウンします。

元のミラー・サーバを新しいプライマリ・サーバとして起動する場合、両方のサーバ上のデータベース・ファイルを同じ状態にする方法が 2 つあります。

- データベースとトランザクション・ログ・ファイルを元のプライマリ・サーバからミラー・サーバにコピーし、ミラー・サーバを新しいプライマリ・サーバとして起動します。ALTER DATABASE 文を使用すると、サーバを強制的にプライマリ・サーバにできます。「ALTER DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- 元のミラー・サーバで、dbbackup を使用してバックアップを実行します。元のプライマリ・サーバにファイルをコピーし、データベース・サーバを起動します。

データベース・ミラーリングとトランザクション・ログ・ファイル

稼働サーバは、起動されると、現在のトランザクション・ログ・ファイルが存在するディレクトリ内のすべてのトランザクション・ログ・ファイルを調べて、適用する必要があるファイルを特定します。データベース・サーバは、トランザクション・ログに記録された操作をデータベースに適用してから、プライマリ・サーバとミラー・サーバのどちらとして動作するかを判断します。

ミラー・ロールを引き受けたサーバは、プライマリ・サーバからトランザクション・ログ・ページの受信を開始します。プライマリでトランザクション・ログの名前が変更された場合は、ミラーでも名前が変更されます。ミラーは、新しいトランザクション・ログ・ページを、トランザクション・ログ用に指定された名前の新しいファイルに書き出します。

プライマリでは、トランザクション・ログ・ファイルを定期的に削除できます。トランザクション・ログ・ファイルの名前が変更されるたび、ミラーには、プライマリで削除されずに残っている最も古いトランザクション・ログ・ファイルがどれであるかが通知されます。ミラーでも、通知されたものより古いトランザクション・ログ・ファイルはすべて削除されます。

トランザクション・ログのトランケートを要求するプライマリ・サーバに対してバックアップを実行している間は、ミラー・サーバが使用できなくなるので、プライマリでのトランザクション・ログの削除は別の方法で実行する必要があります (たとえば、xp_cmdshell を使用して作成後 1 週間が過ぎたファイルを削除するイベントをスケジュールすることで実行します)。

データベース・ミラーリングにおけるシステム・イベント

データベース・ミラーリングでサポートされているシステム・イベントは次のとおりです。

- **MirrorFailover**　このイベントは、データベース・サーバがミラーリングされたデータベースの所有権を取得するたびに起動します。たとえば、初めてサーバが起動され、データベースを所有する必要があると判断した場合に起動します。また、前回ミラーとして動作していたサー

バが、プライマリ・サーバがダウンしており、監視サーバに確認した結果、所有権を取得する必要があると判断した場合にも起動します。

- **MirrorServerDisconnect**　プライマリ・サーバとミラー・サーバまたは監視サーバとの接続が失われると、MirrorServerDisconnect イベントが起動します。このイベントのハンドラ内部で、EVENT_PARAMETER ('MirrorServerName') の値は接続が失われたサーバの名前です。

イベントは、現在ミラー・サーバとして動作しているサーバでは起動しません。また、ミラーリング・イベントを監視サーバ上で実行するよう定義することはできません。これは、イベントは定義されているデータベースのコンテキストでのみ実行されますが、監視サーバはミラーリングされているデータベースのコピーを使用しないからです。

上記のイベントは、ミラー・データベースに対するアクションの必要性を電子メールで通知するためのメカニズムとして使用できます。ただし、プライマリ・サーバ上で実行されているデータベースが使用できなくなるあらゆる状況でこれらのイベントが起動するわけではありません。たとえば、プライマリ・サーバとミラー・サーバの両方に影響する停電があった場合、どちらのイベントも起動しません。このような監視が必要な場合は、別のコンピュータでスクリプト言語を使用することで実装できます。たとえば、dbping を定期的に呼び出してミラー・データベースに接続します。「Ping ユーティリティ (dbping)」 693 ページを参照してください。

次の例は、フェールオーバの発生を管理者に通知するイベントを作成しています。

```
CREATE EVENT mirror_server_unavailable
TYPE  MirrorServerDisconnect
HANDLER
BEGIN
   CALL xp_startmail ( mail_user ='George Smith',
                                                mail_password ='mypwd' );
 CALL xp_sendmail( recipient='DBAdmin',
   subject='Database failover occurred',
   "message"='The following server is unavailable in the mirroring system: '
            || EVENT_PARAMETER( 'MirrorServerName' ) );
   CALL xp_stopmail ( );
END;
```

データベース・ミラーリングとパフォーマンス

プライマリ・サーバとミラー・サーバを実行する各コンピュータは、ハードウェア構成 (プロセッサ、ディスク、メモリなど) を同じにしておくのが理想的です。どちらかのコンピュータで実行されているデータベース・サーバは、常にミラーリング対象のデータベースのプライマリ・サーバとして動作する可能性があります。プライマリでの更新アクティビティによっては、ミラー・サーバの使用率が低くなるのが一般的です。

プライマリ・サーバに対するクエリ・パフォーマンスは、ミラーリングの影響を受けません。データベースを更新するトランザクションのパフォーマンスは、トランザクションのサイズとコミットの頻度の影響を受けます。非同期モードで稼働しているミラー・サーバのパフォーマンスは、同期モードの場合より優れていますが、ミラーリング・システムに参加していないデータベース・サーバよりは悪くなります。パフォーマンスは、稼働サーバ間のネットワーク接続速度に大きく依存します。

データベース・ミラーリングとバックアップ

データベース・ミラーリングはデータ消失のリスクを最低限に抑えるのに役立ちますが、データベース・ミラーリング・システムに参加しているデータベースのバックアップと検証も実施することをおすすめします。

BACKUP DATABASE 文を使用すると、データベース・サーバを基準にしてバックアップを実行できます。BACKUP DATABASE 文はプライマリ・データベース・サーバ上で実行されるので、指定するファイル名には、プライマリおよびミラーのデータベース・サーバ両方で一貫性のあるネットワーク・ドライブまたは UNC 名を指定する必要があります。「BACKUP 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

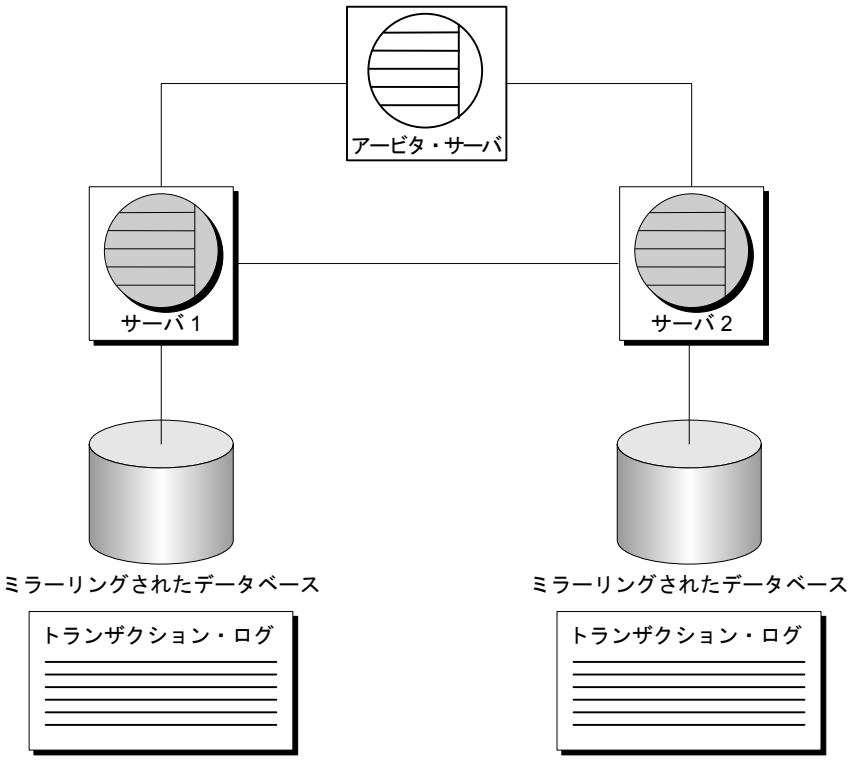
別の方法として、dbbackup ユーティリティを使用してクライアント側のバックアップを実行することもできます。「バックアップ・ユーティリティ (dbbackup)」 640 ページを参照してください。

参照

- 「バックアップとデータ・リカバリ」 811 ページ
- 「データベースの妥当性の確認」 828 ページ

データベース・ミラーリングのシナリオ

以下のシナリオは、ミラーリング・システムでサーバが使用できなくなった場合に何が起こるかを理解するのに役立ちます。ここで使用するデータベース・ミラーリングは、同期モードで実行されるサーバ 1、サーバ 2、監視サーバで構成されています。



ミラーリング・システム内のデータベース・サーバのステータスは、MirrorState、PartnerState、ArbiterState の各データベース・プロパティを使用していつでも確認できます。「データベース・レベルのプロパティ」 550 ページを参照してください。

シナリオ 1：プライマリ・サーバが使用できなくなった場合

1. プライマリ・サーバ (サーバ 1) が使用できなくなります。クライアントはすべて切断されます。
2. 監視サーバとサーバ 2 はサーバ 1 が使用できないことを検知します。
3. 監視サーバとサーバ 2 はクォーラムを形成し、サーバ 2 がプライマリ・サーバになります。
4. サーバ 2 は、クライアント接続の受け付けを開始します。

このシナリオにおいて、非同期モードまたは非同期フルページ・モードを使用し、自動フェールオーバの実行を指定していない場合、クライアントが再び接続できるようにするには、データベースのコピーを作成し、稼働しているサーバを再起動する必要があります。

プライマリ・サーバが使用できなくなったときのリカバリの詳細については、「プライマリ・サーバ障害からのリカバリ」 902 ページを参照してください。

シナリオ 2：プライマリ・サーバが使用できなくなり、再起動された場合

1. 監視サーバとミラー・サーバ (サーバ 2) が、プライマリ・サーバ (サーバ 1) が使用できなくなったことを検知します。

2. 監視サーバとサーバ 2 はクォーラムを形成し、サーバ 2 がプライマリ・サーバになります。
3. サーバ 2 は、クライアント接続の受け付けを開始します。
4. サーバ 1 が再びオンラインになり、サーバ 2 と監視サーバに再接続します。
5. サーバ 1 がクォーラムを要求しますが、サーバ 2 がすでにプライマリ・サーバになっています。
6. サーバ 1 はミラー・サーバになって、サーバ 2 からの変更を待機します。
7. サーバ 2 は変更をサーバ 1 に送信します。

サーバ 1 がサーバ 2 からすべてのトランザクションを受信する前にサーバ 2 が使用できなくなった場合、サーバ 1 は同期された状態にはなれません。サーバ 2 が再び使用可能になるのを待って、まだ取得していないトランザクションを取得して適用する必要があります。

プライマリ・サーバが使用できなくなったときのリカバリの詳細については、「プライマリ・サーバ障害からのリカバリ」 902 ページを参照してください。

シナリオ 3：ミラー・サーバが使用できなくなった場合

1. ミラー・サーバ (サーバ 2) が使用できなくなります。
2. 監視サーバとサーバ 1 はミラー・サーバ (サーバ 2) が使用できないことを検知します。

   クライアント接続は影響を受けません。引き続きプライマリ・サーバに接続できます。ただし、サーバ 1 または監視サーバが使用できなくなった場合、クライアントは接続できなくなります。

シナリオ 4：ミラー・サーバが使用できなくなり、再起動された場合

1. ミラー・サーバ (サーバ 2) が使用できなくなります。
2. 可用性は変わらないため、クライアント接続は影響を受けません。引き続きプライマリ・サーバに接続できます。ただし、サーバ 1 または監視サーバが使用できなくなった場合、クライアントは接続できなくなります。
3. サーバ 2 が再びオンラインになり、サーバ 1 と監視サーバに再接続します。
4. サーバ 2 がクォーラムを要求しますが、サーバ 1 がすでにプライマリ・サーバになっています。
5. サーバ 2 はミラー・サーバになって、サーバ 1 からの変更を待機します。
6. サーバ 1 は変更をサーバ 2 に送信します。

   可用性は変わらないため、クライアント接続は影響を受けません。引き続きサーバ 1 に接続します。

シナリオ 5：監視サーバが使用できなくなった場合

1. サーバ 1 (プライマリ・サーバ) とサーバ 2 (ミラー・サーバ) が、監視サーバがダウンしたことを検知します。
2. どちらのサーバも引き続き使用できます。クライアントは切断されません。

監視サーバが再びオンラインになると、サーバ 1 とサーバ 2 はそれを検知して、監視サーバとの通信を開始します。クライアントにとっては、データベースの可用性に変化はありません。

監視サーバがない状態でサーバ 1 またはサーバ 2 が使用できなくなった場合、もう一方のサーバは単独ではクォーラムを満たすことができないので、データベースは使用できなくなります。

シナリオ 6：監視サーバが再起動された場合

1. 監視サーバが再びオンラインになり、サーバ 1 とサーバ 2 に再接続します。

   可用性は変わらないため、クライアント接続は影響を受けません。

SQL Anywhere Veritas Cluster Server エージェントの使用

「**クラスタ**」とは、一連のアプリケーションを実行するために連携して動くコンピュータ (**ノード**と呼ばれる) によるグループです。クラスタ上で実行されているアプリケーションに接続するクライアントは、クラスタを単一のシステムとして扱います。あるノードに障害が発生すると、クラスタ内の別のノードが、障害のあったノードが提供するサービスを自動的に引き継ぎます。クライアント側では、可用性がわずかに悪化したように見えることがありますが (残りのノードでサービスを再開するまでの時間)、ノードに障害が発生したことまではわかりません。

SQL Anywhere でクラスタリングを使用する場合、データベースまたはデータベース・サーバがクラスタ内の別のノードにフェールオーバされると、コミットされていないトランザクションはすべて失われます。また、フェールオーバが発生した場合は、クライアントはデータベースに再接続する必要があります。

SQL Anywhere ではさまざまなクラスタ環境をサポートしています。そのような環境では、クラスタ・ソフトウェアが任意のアプリケーションを自動フェールオーバの対象となる汎用リソースにすることで、高可用性が実現されます。ただし、フェールオーバできるのはデータベース・サーバ・プロセスのみで、監視プロセスや制御プロセスについては制限があります。

詳細については、http://www.ianywhere.jp/developers/technotes/asa_cluster_db_service.html を参照してください。

ほとんどのクラスタ・ソフトウェアには、特定のアプリケーション用のカスタム・リソースを作成するための API が用意されています。SQL Anywhere には、Veritas Cluster Server 用のカスタム・フェールオーバ・リソースとして、SAServer と SADatabase の 2 つが用意されています。SAServer エージェントはデータベース・サーバのフェールオーバを担当し、SADatabase エージェントは個々のデータベース・ファイルのフェールオーバを担当します。エージェントは、アプリケーションに応じて、いずれか一方または両方とも使用できます。

SQL Anywhere Veritas Cluster Server エージェントを使用するには、次のようにシステムを設定する必要があります。

- Veritas Cluster Server 4.1 以降を使用する
- クラスタ内の各システムに SQL Anywhere を同じ構成でインストールする
- データベース・ファイルが、クラスタ内のすべてのシステムからアクセスできる共有記憶装置に格納されている
- ユーティリティ・データベースのパスワードが、クラスタ内のすべてのシステムで同じである

  SADatabase エージェントはユーティリティ・データベースを使用して、特定のデータベース・ファイルを起動および停止します。クラスタに属しているすべてのシステムには、同じユーティリティ・データベース・パスワードを使用する必要があります。ユーティリティ・データベースのパスワードを設定するには、データベース・サーバの起動時に -su サーバ・オプションを指定するか、クラスタの各ノードの *install-dir¥win32¥util_db.ini* ファイルに次の行を追加します。

pwd = [*password-of-your-choice*]

UNIX の場合、VCS エージェントは *install-dir/vcsagent/saserver* にインストールされます。

新しいエージェントを設定して Veritas Cluster Server に追加するには、次の 3 つの方法があります。

1. Cluster Manager を使用する。
2. コマンド・ライン・ユーティリティを使用する。
3. テキスト・エディタを使用して、*main.cf* 設定ファイルを編集する。

ここでは、Cluster Manager を使用する手順について説明します。

使用可能なユーティリティの詳細については、『*Veritas Cluster Server Administration Guide*』を参照してください。

main.cf をテキスト・エディタを使用して手動で設定する場合は、*main.cf* ファイルを編集する前にすべての Veritas Cluster Server サービスを停止する必要があります。そうしないと、変更が反映されません。

SAServer エージェントの設定

SAServer エージェントは、SQL Anywhere データベース・サーバによるクラスタ内の別のノードへのフェールオーバを制御します。

♦ **SAServer エージェントを設定するには、次の手順に従います。**

1. クラスタの各ノードで実行されている SQL Anywhere データベース・サーバをすべてシャットダウンします。
2. クラスタからノードを選択して、*SAServer* というディレクトリをそのノードの *%VCS_HOME%¥bin* ディレクトリの下に作成します。他の Veritas Cluster Server エージェントがこのフォルダに作成されます (NIC や IP など)。
3. 次のファイルを *install-dir¥VCSAgent¥SAServer* ディレクトリから手順 2 で作成した *SAServer* ディレクトリにコピーします。
   - *Online.pl*
   - *Offline.pl*
   - *Monitor.pl*
   - *Clean.pl*
   - *SAServer.xml*
4. ファイル *%VCS_HOME%¥bin¥VCSdefault.dll* を *%VCS_HOME%¥bin¥SAServer* ディレクトリにコピーし、名前を *SAServer.dll* に変更します。
5. ファイル *install-dir¥VCSAgent¥SAServer¥SAServerTypes.cf* を *%VCS_HOME%¥conf¥config* ディレクトリにコピーします。

6. クラスタ内のその他すべてのノードについて、手順 1 ー 5 を繰り返します。
7. Veritas Cluster Server Manager を起動し、ユーザ名とパスワードを入力してクラスタに接続します。
8. 次の手順で SAServer エージェントを追加します。
   a. [File] − [Import Types] を選択します。
   b. *%VCS_HOME%¥conf¥config¥SAServerTypes.cf* に移動して、[Import] をクリックします。

♦ SAServer エージェントを使用してフェールオーバするデータベース・サーバを設定するには、次の手順に従います。

1. Veritas Cluster Server Manager を起動し、ユーザ名とパスワードを入力して接続します。
2. 次の手順で、SAServer をリソースとしてサービス・グループに追加します。
   a. [Edit] − [Add] − [Resource] を選択します。

      [Add Resource] ダイアログが表示されます。
   b. [Resource Type] ドロップダウン・リストから SAServer を選択します。

      Windows では、[Resource Type] リストで Windows の下に SAServer が表示されない場合、*SAServer.xml* ファイルを *%VCS_ROOT%¥cluster manager¥attrpool¥Win2K¥400* に追加して、クラスタ・サービスを再起動する必要があります。
   c. [Resource Name] フィールドに名前を入力します。
   d. 以下の属性に次のように属性値を追加します。
      - **cmdStart** dbsrv10 -x tcpip *database-file-on-shared-disk* -n *server-name*
      - **cmdMonitor** dbping -c "ENG=*server-name*"
      - **cmdStop** dbstop -c *user-id*,*password* -y
   e. [Enabled] を選択します。

      これにより、リソースの使用準備が整ったことを示します。
   f. [OK] をクリックします。
3. リソースの依存性が適切に設定されていることを確認します。共有ディスク・リソースや IP アドレス・リソースなど、SAServer を起動するために起動してグループ化しておく必要があるリソースが他にも存在します。
4. サービス・グループを右クリックして、[Online] − [*node-name*] を選択します。*node-name* は、リソースの実行に使用する、クラスタ内のマシンの名前です。

   これで、サービス・グループがオンラインになります。

SAServer エージェントのテスト

次の手順では、SAServer エージェントのフェールオーバをテストする方法について説明します。

♦ **SAServer エージェントのフェールオーバをテストするには、次の手順に従います。**

1. Interactive SQL からデータベースに接続します。次に例を示します。

   ```
   dbisql -c "UID=DBA;PWD=sql;ENG=VCS;LINKS=tcpip"
   ```

2. 次のクエリを実行します。

   ```
   SELECT * FROM Departments;
   ```

   このクエリはエラーなく実行されます。

3. データベース・サーバを実行しているシステムをシャットダウンします。

   これにより、フェールオーバが発生し、すべてのリソースが代替サーバで起動されます。

4. 同じ接続文字列を使用して Interactive SQL に再接続し、クエリを再度実行します。接続もクエリの実行も正常に行われます。

SADatabase エージェントの設定

SADatabase エージェントは、SQL Anywhere データベースによるクラスタ内の別のノードへのフェールオーバを制御します。

♦ **SADatabase エージェントを設定するには、次の手順に従います。**

1. クラスタの各ノードで実行されている SQL Anywhere データベース・サーバをすべてシャットダウンします。
2. クラスタ内のいずれか 1 つのノードに *%VCS_HOME%¥bin¥SADatabase* というディレクトリを作成します。
3. 次のファイルを *install-dir¥SADatabase* ディレクトリから手順 2 で作成した *%VCS_HOME%¥bin¥SADatabase* ディレクトリにコピーします。

   - *Online.pl*
   - *Offline.pl*
   - *Monitor.pl*
   - *Clean.pl*
   - *SADatabase.xml*

4. ファイル *%VCS_HOME%¥bin¥VCSdefault.dll* を *%VCS_HOME%¥bin¥SADatabase* ディレクトリにコピーし、名前を *SADatabase.dll* に変更します。
5. ファイル *install-dir¥SADatabase¥SADatabaseTypes.cf* を *%VCS_HOME%¥conf¥config* ディレクトリにコピーします。
6. クラスタに属するその他すべてのシステムについて、手順 1 ー 5 を繰り返します。

7. Veritas Cluster Server Manager を起動し、ユーザ名とパスワードを入力してクラスタに接続します。
8. 次の手順で SADatabase エージェントを追加します。
   a. [File] − [Import Types] を選択します。
   b. *%VCS_HOME%¥conf¥config¥* に移動して、[Import] をクリックします。

♦ SADatabase エージェントを使用してフェールオーバするデータベースを設定するには、次の手順に従います。

1. 次の手順で、SADatabase をリソースとしてサービス・グループに追加します。
   a. [Edit] − [Add] − [Resource] を選択します。

      [Add Resource] ダイアログが表示されます。
   b. [Resource Type] ドロップダウン・リストから SADatabase を選択します。

      Windows では、[Resource Type] リストで Windows の下に SADatabase が表示されない場合、*SADatabase.xml* ファイルを *%VCS_ROOT%¥cluster manager¥attrpool¥Win2K¥400* に追加して、クラスタ・サービスを再起動する必要があります。
   c. [Resource Name] フィールドに名前を入力します。
   d. 以下の各属性について、[Edit] 列のボタンをクリックして、次のように属性値を追加します。
      - **DatabaseFile**　データベース・ファイルのロケーション。たとえば、*E:¥demo.db*。
      - **DatabaseName**　データベースの名前。
      - **ServerName**　データベース・サーバの名前。クラスタ内の各システムには異なるサーバ名を指定できます。属性のスコープは、Global ではなく、Per System にします。
      - **UtilDBpwd**　クラスタ内のすべてのシステムで使用されるユーティリティ・データベースのパスワード。
   e. [Enabled] を選択します。

      これにより、リソースの使用準備が整ったことを示します。
   f. [OK] をクリックします。
2. リソースの依存性が適切に設定されていることを確認します。共有ディスク・リソースや IP アドレス・リソースなど、SADatabase を起動するために起動/グループ化しておく必要があるリソースが他にも存在します。
3. サービス・グループを右クリックして、[Online] − [*node-name*] を選択します。*node-name* は、リソースの実行に使用する、クラスタ内のマシンの名前です。

   これで、サービス・グループがオンラインになります。

SADatabase エージェントのテスト

次の手順では、SADatabase エージェントのフェールオーバをテストする方法について説明します。

♦ **SADatabase エージェントのフェールオーバをテストするには、次の手順に従います。**

1. Interactive SQL からデータベースに接続します。次に例を示します。

   ```
   dbisql -c "UID=DBA;PWD=sql;ENG=VCS;LINKS=tcpip"
   ```

2. 次のクエリを実行します。

   ```
   SELECT * FROM Departments;
   ```

   このクエリはエラーなく実行されます。

3. データベースに障害が発生し、第 1 システム・ノードで実行されているデータベース・サーバがデータベース・ファイルにアクセスできないとします。これにより、第 2 システム・ノードで起動されたデータベース・サーバへのデータベース・ファイルのフェールオーバが発生します。第 1 ノード上のデータベース・ファイルに障害を発生させるには、次のようなコマンドを発行します。

   ```
   dbisql -q -c "UID=DBA;PWD=sql;ENG=VCS1;DBN=utility_db" STOP DATABASE DEMO ON VCS1
   UNCONDITIONALLY;
   ```

   第 1 コンピュータ上のデータベース・ファイルに障害が発生します。Veritas Cluster Server がこのファイルに障害が発生したことを認識するまでには遅延があります。これは、Veritas Cluster Server がリソースの状態を監視する間隔がデフォルトでは 60 秒ごとになっているからです (この間隔は、リソース設定で短くできます)。これにより、データベース・ファイルは第 2 コンピュータにフェールオーバされ、データベース・ファイルは第 2 コンピュータ上のデータベース・サーバを使用して起動されます。このデータベース・サーバの名前は、元のデータベース・サーバとは名前が異なる場合があります。

   たとえば、新しいデータベース・サーバが VCS2 の場合、クライアントでは次のようにして接続文字列にこの新しいデータベース・サーバ名を指定する必要があります。

   ```
   "UID=DBA;PWD=sql;ENG=VCS2;DBN=DEMO;LINKS=tcpip"
   ```

4. Interactive SQL に再接続します。接続もクエリの実行も正常に行われます。

パート VI. セキュリティ

パート VI では、SQL Anywhere のセキュリティ機能について説明します。

第 23 章

安全なデータの管理

目次

セキュリティ機能の概要

データベースには機密の情報や個人的な情報などが含まれている場合があるので、データベースやそこに含まれるデータのセキュリティを考慮した設計になっていることが重要です。

SQL Anywhere には、データの安全な環境の構築に役立ついくつかの機能があります。

- **ユーザ ID と認証**　データベースにアクセスするユーザを制御します。「新しいユーザの作成」 374 ページを参照してください。
- **任意アクセス制御機能**　データベースへの接続中にユーザが実行するアクションを制御します。「データベースのパーミッションの概要」 370 ページを参照してください。
- **監査**　データベースで行われたアクションの記録を管理するのに役立ちます。「データベース・アクティビティの監査」 930 ページを参照してください。
- **データベース・サーバ・オプション**　データベースのロードなどの管理作業を実行するユーザを指定します。このオプションは、データベース・サーバの起動時に設定されます。「コマンド・ラインからパーミッションを制御する」 22 ページを参照してください。
- **ビューとストアド・プロシージャ**　ユーザがアクセスするデータとユーザが実行する操作を指定します。「高度なセキュリティを実現するためのビューとプロシージャの使い方」 396 ページを参照してください。
- **データベースとテーブルの暗号化**　データベースを安全に管理するために、単純暗号化または強力な暗号化のいずれかを選択できます。単純暗号化は、難読化と同じです。強力な暗号化にすると、暗号化キーなしではデータベースにまったくアクセスできなくなります。「-ek データベース・オプション」 222 ページと「DatabaseKey 接続パラメータ [DBKEY]」 242 ページを参照してください。

  テーブル暗号化機能では、データベース全体ではなく個々のテーブルを暗号化できます。「テーブル暗号化」 943 ページを参照してください。
- **トランスポート・レイヤ・セキュリティ**　トランスポート・レイヤ・セキュリティを使用すると、クライアント・アプリケーションとデータベース・サーバ間の通信を認証することができます。トランスポート・レイヤ・セキュリティでは、楕円曲線暗号方式または RSA 暗号方式を使用します。「トランスポート・レイヤ・セキュリティ」 949 ページを参照してください。

> **注意**
> データベース・サーバを実行しているコンピュータ上で他のプロセスがクライアント／サーバ通信の内容にアクセスできるようになることが心配な場合は、暗号化の使用をおすすめします。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」『SQL Anywhere 10 - 紹介』を参照してください。

- **機能の保護** データベース・サーバ上のすべてのデータベースで機能を無効にすることができます。

データのセキュリティについては、データベース管理者に責任があります。この章で記述されているタスクを実行するには、特に明記されていないかぎり、DBA 権限が必要です。

ユーザ ID とパーミッションはセキュリティ関連のトピックです。「ユーザ ID とパーミッションの管理」 369 ページを参照してください。

セキュリティのヒント

データベース管理者には、データのセキュリティ強化のために実行するアクションが多数用意されています。次に例を示します。

- **デフォルトのユーザ ID とパスワードの変更**　新しく作成されるデータベースのデフォルトのユーザ ID とパスワードは、**DBA** と **sql** です。このパスワードを変更してから、データベースを展開してください。
- **長いパスワードが必要**　min_password_length パブリック・オプションを設定して、短いパスワードを指定できないように (つまり簡単に推測できないように) することができます。「min_password_length オプション [データベース]」 470 ページを参照してください。
- **DBA 権限の制限**　DBA 権限は非常に強力であるため、本当に必要なユーザにだけ付与するようにしてください。DBA 権限を持っているユーザは、データベース内で何でも参照したり実行したりできます。

  DBA 権限を持つユーザには、ユーザ ID を 2 つ与えてください。1 つを DBA 権限付き、もう 1 つを DBA 権限なしにすれば、必要なときにだけ DBA として接続できます。
- **保護されたデータベース機能の使用**　-sf データベース・サーバ・オプションでは、データベース・サーバ上のすべてのデータベースに対して機能を有効または無効にすることができます。無効にできる機能には、外部ストアド・プロシージャや Java の使用、リモート・データ・アクセス、要求ログ設定の変更機能などがあります。「-sf サーバ・オプション」 192 ページと「保護された機能の指定」 928 ページを参照してください。
- **外部システム関数の削除**　外部関数の xp_cmdshell、xp_startmail、xp_startsmtp、xp_sendmail、xp_stopmail、xp_stopsmtp は、セキュリティを考慮した環境で使用すると問題が発生する可能性があります。

  xp_cmdshell プロシージャを使用すると、ユーザはオペレーティング・システム・コマンドやプログラムを実行できます。

  電子メール・コマンドを使用すると、ユーザはサーバに自分が作成した電子メールを送信させることができます。悪意のあるユーザであれば、電子メール・コマンドやコマンド・シェル・プロシージャを使用して、付与されていない権限でオペレーティング・システムのタスクを実行することもありえます。セキュリティを考慮した環境では、このような関数は削除してください。

  プロシージャの削除については、「DROP 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **データベース・ファイルの保護**　データベース・ファイル、ログ・ファイル、DB 領域ファイルを不正アクセスから保護する必要があります。このようなファイルは共有ディレクトリまたはボリュームには保管しないでください。
- **データベース・ソフトウェアの保護**　SQL Anywhere ソフトウェアも同様に保護する必要があります。ユーザには、アプリケーション、DLL、その他の必要なリソースへのアクセス権だけを付与してください。

- **サービスまたはデーモンとしてのデータベース・サーバの実行**　権限のないユーザがサーバを停止したり、データベースやログ・ファイルへのアクセスを取得したりしないように、データベース・サーバを Windows サービスとして実行してください。UNIX では、同様の目的でサーバをデーモンとして実行します。「現在のセッション外でのサーバの起動」 36 ページを参照してください。

- **SATMP をユニークなディレクトリに設定**
  UNIX プラットフォームでデータベース・サーバを保護するには、SATMP をユニークなディレクトリに設定し、他のすべてのユーザに対して、このディレクトリの読み取り、書き込み、実行を制限します。これによって、すべての接続が強制的に TCP/IP を使用することになり、共有メモリ接続よりも安全になります。

  クライアントとサーバ間で使用される共有メモリのバッファは、実際のデータが 2 つのサイト間で送受信される前に、ディレクトリ・ツリーから削除されます。これは、共有メモリのバッファやファイルが非表示であるため、他のプロセスは通信データを参照できず、データを処理できないことを意味します。

- **データベースを強力に暗号化**　データベースを強力に暗号化すると、キーがなければまったくアクセスできなくなります。他の方法を使っても、データベースを開いたり、データベースやトランザクション・ログ・ファイルを表示したりできません。

  詳細については、「-ep サーバ・オプション」 164 ページおよび 「-ek データベース・オプション」 222 ページを参照してください。

データベース・アクセスの制御

データベース管理者は、ユーザ ID とパスワードを割り当てることによって、どのユーザがデータベースにアクセスするかを制御します。各ユーザ ID にパーミッションを付与することによって、データベース接続しているときに各ユーザが実行できるタスクを制御します。

ユーザ ID に基づくパーミッションのスキーム

ユーザは、データベースにログインすると、次の基準の*いずれか*を満たすすべてのデータベース・オブジェクトにアクセスできます。

- ユーザが作成したオブジェクト
- ユーザに対して明示的なパーミッションが付与されているオブジェクト
- ユーザの所属グループに明示的なパーミッションが付与されているオブジェクト

ユーザは、この基準を満たさないデータベース・オブジェクトにはアクセスできません。つまりユーザは、自分が所有するオブジェクト、またはアクセス権限を明示的に付与されているオブジェクトにのみアクセスできます。

詳細については、次の項を参照してください。

- 「ユーザ ID とパーミッションの管理」 369 ページ
- 「CONNECT 文 [ESQL] [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』
- 「GRANT 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「REVOKE 文」『SQL Anywhere サーバ - SQL リファレンス』

統合化ログインの使用方法

統合化ログインを使用すると、ユーザは 1 つのログイン名とパスワードで Windows オペレーティング・システムとデータベースの両方にログインできます。外部ログイン名は、データベース・ユーザ ID に関連付けられています。統合化ログインを行う場合、ユーザはログイン名とパスワードの両方を指定してオペレーティング・システムにログオンします。オペレーティング・システムはそのユーザをサーバに通知し、サーバは関連付けられたデータベース・ユーザ ID としてそのユーザをログインさせます。追加のログイン名とパスワードは必要ありません。

統合化ログインを使用する場合、ユーザ・プロファイル Guest でブランクのパスワードを使用できるようにしておくと、統合化ログインを受け入れるサーバで管理しているデータベースへのアクセスが無制限に許可されます。ユーザはデフォルトではログイン時に Guest ユーザのプロファイルを使用するため、現実には、どのユーザでも任意のログイン ID やパスワードを使用してサーバにログインできてしまいます。

詳細については、次の項を参照してください。

- 「セキュリティについての考慮事項：無制限データベース・アクセス」 106 ページ
- 「統合化ログインの使用方法」 98 ページ
- 「login_mode オプション [データベース]」 459 ページ

パスワードのセキュリティの強化

パスワードは、データベースのセキュリティ・システムの重要な部分です。安全のために、パスワードは容易に推測できないものにし、ハード・ドライブやその他のロケーションから簡単にアクセスできないようにしてください。SQL Anywhere のパスワードでは、常に大文字と小文字が区別されます。パスワード認証に使用する関数を verify_password_function オプションで指定できます。「verify_password_function オプション [データベース]」 513 ページを参照してください。

最小長のパスワードの実装

デフォルトでは、パスワードは任意の長さで指定できます。セキュリティを強化するために、新しいパスワードに必要な最小長を課すことができます。これを実行するには、min_password_length データベース・オプションを 0 より大きな値に設定します。次の文は、パスワードが最低でも 8 バイトの長さになるようにします。

```
SET OPTION PUBLIC.min_password_length = 8;
```

「min_password_length オプション [データベース]」 470 ページを参照してください。

パスワード有効期限の導入

デフォルトでは、データベース・パスワードに有効期限はありません。post_login_procedure オプションを使用することによって、パスワードの期限切れが近づいたことをユーザに知らせ、パスワードを変更するように警告するなどの機能を実装できます。「post_login_procedure [データベース]」 481 ページを参照してください。

ODBC データ・ソースにパスワードを含めない

パスワードはデータベースへのアクセスのキーとなります。そのため、セキュリティを考慮した環境では、権限のないユーザが簡単にパスワードを使用できないようにすることが重要です。

ODBC データ・ソースを作成するとき、または Sybase Central 接続プロファイルを作成するときに、オプションでパスワードを含めることができます。権限のないユーザによって参照されないようにするため、パスワードは含めないようにしてください。

「ODBC データ・ソースの作成」 76 ページを参照してください。

パスワードを含む設定ファイルの暗号化

設定ファイルを作成するとき、パスワード情報をオプションとして組み込むことができます。パスワードを保護するために、ファイル非表示 (dbfhide) ユーティリティを使用し、単純暗号化で設定ファイルの内容を隠すことを検討してください。「ファイル非表示ユーティリティ (dbfhide)」 657 ページを参照してください。

パスワード検証関数の使用

verify_password_function オプションを使用して、パスワード規則を実装する関数を指定できます。「verify_password_function オプション [データベース]」 513 ページを参照してください。

次に示す例では、プロシージャと関数の数を定義します。同時に、パスワードに特定の種類の文字を要求し、パスワードの再利用を禁止して、パスワード有効期限を適用するなど、詳細なパスワード規則を実装しています。これらのプロシージャと関数は、ユーザ ID の作成、パスワード

の変更、ユーザの接続が行われたときに、verify_password_function オプションと login_procedure オプションを使用してデータベース・サーバから呼び出されます。アプリケーションは、post_login_procedure オプションで指定されたプロシージャを呼び出して、パスワードが期限切れになる前にパスワードの変更が必要であることを報告できます。

このサンプル・コードは、*samples-dir¥SQLAnywhere¥SQL¥verify_password.sql* でも利用できます。*samples-dir* の詳細については、「サンプル・ディレクトリ」 323 ページを参照してください。

```
-- only DBA should have permissions on this table
CREATE TABLE DBA.t_pwd_history(
       pk          INT         DEFAULT AUTOINCREMENT PRIMARY KEY,
       change_date TIMESTAMP   DEFAULT CURRENT TIMESTAMP,  -- when pwd set
       grace_date  DATE,               -- a day after password expires to
                                       -- allow user to log in
       user_name   CHAR(128),          -- the user whose password is set
       pwd_hash    CHAR(32) );         -- hash of password value to detect
                                       -- duplicate passwords

-- called on every GRANT CONNECT TO ... IDENTIFIED BY ... statement
-- to verify the password conforms to password rules
CREATE FUNCTION DBA.f_verify_pwd( uid     VARCHAR(128),
                                  new_pwd VARCHAR(255) )
RETURNS VARCHAR(255)
BEGIN
   -- a table with one row per character in new_pwd
   DECLARE local temporary table pwd_chars(
          pos INT PRIMARY KEY,    -- index of c in new_pwd
          c   CHAR( 1 CHAR ) );   -- character
   -- new_pwd with non-alpha characters removed
   DECLARE pwd_alpha_only      CHAR(255);
   DECLARE num_lower_chars     INT;

   -- enforce minimum length (can also be done with
   -- min_password_length option)
   IF LENGTH( new_pwd ) < 6 THEN
      RETURN 'password must be at least 6 characters long';
   END IF;

   -- break new_pwd into one row per character
   INSERT INTO pwd_chars SELECT row_num, SUBSTR( new_pwd, row_num, 1 )
                         FROM dbo.RowGenerator
                         WHERE row_num <= LENGTH( new_pwd );

   -- copy of new_pwd containing alpha-only characters
   SELECT LIST( c, '' ORDER BY pos ) INTO pwd_alpha_only
      FROM pwd_chars WHERE c BETWEEN 'a' AND 'z' OR c BETWEEN 'A' AND 'Z';

   -- number of lower case characters IN new_pwd
   SELECT COUNT(*) INTO num_lower_chars
      FROM pwd_chars WHERE CAST( c AS BINARY ) BETWEEN 'a' AND 'z';

   -- enforce rules based on characters contained in new_pwd
   IF ( SELECT COUNT(*) FROM pwd_chars WHERE c BETWEEN '0' AND '9' )
        < 1 THEN
      RETURN 'password must contain at least one numeric digit';
   ELSEIF LENGTH( pwd_alpha_only ) < 2 THEN
      RETURN 'password must contain at least two letters';
   ELSEIF num_lower_chars = 0
         OR LENGTH( pwd_alpha_only ) - num_lower_chars = 0 THEN
      RETURN 'password must contain both upper- and lowercase characters';
   END IF;
```

```
    -- not the same as any user name
    -- (this could be modified to check against a disallowed words table)
    IF EXISTS( SELECT * FROM SYS.SYSUSER
                WHERE LOWER( user_name ) IN ( LOWER( pwd_alpha_only ),
                                              LOWER( new_pwd ) ) ) THEN
       RETURN 'password or only alphabetic characters in password ' ||
              'must not match any user name';
    END IF;

    -- not the same as any previous password for this user
    IF EXISTS( SELECT * FROM t_pwd_history
                WHERE user_name = uid
                  AND pwd_hash = HASH( uid || new_pwd, 'md5' ) ) THEN
       RETURN 'previous passwords cannot be reused';
    END IF;

    -- save the new password
    INSERT INTO t_pwd_history( user_name, pwd_hash )
       VALUES( uid, HASH( uid || new_pwd, 'md5' ) );

    RETURN( NULL );
END;

ALTER FUNCTION DBA.f_verify_pwd SET HIDDEN;
GRANT EXECUTE ON DBA.f_verify_pwd TO PUBLIC;
SET OPTION PUBLIC.verify_password_function = 'DBA.f_verify_pwd';

-- called on every connection to check for password expiry
CREATE PROCEDURE DBA.p_login_check()
BEGIN
    DECLARE uid                CHAR(128);
    DECLARE INVALID_LOGON      EXCEPTION FOR SQLSTATE '28000';
    DECLARE last_pwd_change    DATE;
    DECLARE grace_date         DATE;
    DECLARE is_dba             CHAR;
    DECLARE msg_str            CHAR(255);

    SET uid = CONNECTION_PROPERTY( 'Userid' );
    IF ( EXISTS( SELECT * FROM t_pwd_history WHERE user_name = uid ) ) THEN
       SELECT FIRST  t.change_date, t.grace_date
              INTO last_pwd_change, grace_date
              FROM t_pwd_history t WHERE t.user_name = uid
              ORDER BY t.change_date DESC;
    END IF;
    IF last_pwd_change IS NULL THEN
       -- no password change in t_pwd_history, so create one now.
       INSERT INTO t_pwd_history( user_name, pwd_hash )
          VALUES( uid, 'unknown' );
       COMMIT WORK;
    ELSE
       IF EXISTS( SELECT * FROM SYS.SYSUSERAUTHORITY a, SYS.SYSUSER u
                  WHERE u.user_name = uid AND u.user_id = a.user_id AND
                        a.auth = 'DBA' ) THEN
          SET is_dba = 'Y';
       ELSE
          SET is_dba = 'N';
       END IF;

       -- remove any locks on t_pwd_history and SYSUSERAUTHORITY
       ROLLBACK WORK;
       -- check if last password change was over five months ago
       IF CURRENT DATE > DATEADD( month, 5, last_pwd_change ) THEN
          -- Never expire DBA accounts so that the database does not
          -- get locked out by all users.
```

```
            IF CURRENT DATE < DATEADD( month, 6, last_pwd_change ) OR
              is_dba = 'Y' OR
              ( grace_date IS NOT NULL AND grace_date = CURRENT DATE ) THEN
               SET msg_str = 'The password for user ' || uid ||
                           ' expires on ' ||
                           CAST( DATEADD( month, 6, last_pwd_change )
                                AS DATE ) ||
                           '.  Please change it before it expires.';
               MESSAGE msg_str;
               -- The post_login_procedure option is set to
               -- p_post_login_check, which will cause a dialog to be
               -- displayed notifying the user their password will
               -- expire soon. dbisql and dbisqlc will display this
               -- dialog, and user applications can call the
               -- post_login_procedure and display this dialog.
               -- May want to use xp_send_mail to notify user and/or
               -- administrator.
            ELSEIF grace_date IS NULL THEN
               -- Allow one grace login day. The first login on the grace
               -- day fails to ensure the user knows their password has
               -- expired
               UPDATE t_pwd_history t SET t.grace_date = CURRENT DATE
                  WHERE t.grace_date IS NULL AND t.user_name = uid;
               COMMIT WORK;
               SET msg_str = 'The password for user ' || uid ||
                           ' has expired, but future logins will ' ||
                           'be allowed today only so that the password ' ||
                           'can be changed.';
               MESSAGE msg_str;
               RAISERROR 28000 msg_str;
               RETURN;
            ELSE
               SET msg_str = 'The password for user ' || uid ||
                           ' has expired and must be reset by your DBA.';
               MESSAGE msg_str;
               -- may want to use xp_send_mail to notify administrator.
               RAISERROR 28000 msg_str;
               RETURN;
            END IF;
         END IF;
      END IF;

      CALL sp_login_environment;
   END;

   GRANT EXECUTE ON DBA.p_login_check TO PUBLIC;
   SET OPTION PUBLIC.login_procedure = 'DBA.p_login_check';

   -- called by dbisql, dbisqlc and some user applications on every successful
   -- connection to check for warnings which should be displayed
   CREATE PROCEDURE DBA.p_post_login_check()
   RESULT( warning_text VARCHAR(255), warning_action INT )
   BEGIN
      DECLARE uid             CHAR(128);
      DECLARE last_pwd_change DATE;
      DECLARE warning_text    CHAR(255);
      DECLARE warning_action  INT;

      SET uid = CONNECTION_PROPERTY( 'Userid' );
      SELECT FIRST t.change_date
            INTO last_pwd_change
            FROM t_pwd_history t WHERE t.user_name = uid
            ORDER BY t.change_date DESC;
      IF CURRENT DATE > DATEADD( month, 5, last_pwd_change ) THEN
```

```
        SET warning_text = 'Your password expires on ' ||
                        CAST( DATEADD( month, 6, last_pwd_change )
                            AS DATE ) ||
                        '.  Please change it before it expires.';
        SET warning_action = 1;
    ELSE
        -- There is no warning
        SET warning_text = NULL;
        SET warning_action = 0;
    END IF;
    -- Return the warning (if any) through this result set
    SELECT warning_text, warning_action;
END;

GRANT EXECUTE ON DBA.p_post_login_check TO PUBLIC;
SET OPTION PUBLIC.post_login_procedure = 'DBA.p_post_login_check';
```

ユーザが実行できるタスクの制御

ユーザは、アクセスを付与されたオブジェクトにだけアクセスできます。

オブジェクトに対するパーミッションを別のユーザに付与するには、GRANT 文を使用します。オブジェクトに対する権限を付与するパーミッションを、他のユーザにデリゲートすることもできます。

GRANT 文は、より一般的なパーミッションをユーザに付与する場合にも使用します。

- ユーザに CONNECT パーミッションを付与すると、データベースに接続することができます。
- RESOURCE 権限が付与されたユーザは、テーブル、ビュー、プロシージャなどを作成できます。
- DBA 権限が付与されたユーザは、データベース内で何でも参照したり実行したりできます。DBA は、グループの作成と管理にも GRANT 文を使用します。

REVOKE 文は、GRANT 文と反対の機能を持っています。GRANT によって明示的に付与されたパーミッションは、REVOKE によって取り消されます。ユーザから CONNECT を取り消すと、そのユーザは、所有するすべてのオブジェクトとともにデータベースから削除されます。

ネガティブ・パーミッション

SQL Anywhere は「**ネガティブ・パーミッション**」をサポートしません。つまり、明示的に付与されていないパーミッションは取り消せません。

たとえば、ユーザ bob が sales グループのメンバであるとします。あるユーザがテーブル T に対する DELETE パーミッションを sales に付与すると、bob は T からローを削除できるようになります。bob に T からの削除を実行させないようにする場合は、REVOKE DELETE を単純に実行しただけでは bob から T へのパーミッションを取り消すことはできません。T に対する DELETE パーミッションは、直接 bob に付与されたものではないためです。この場合は、sales グループの bob のメンバシップを取り消さなければなりません。

次の項を参照してください。

- 「GRANT 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「REVOKE 文」『SQL Anywhere サーバ - SQL リファレンス』

セキュリティを考慮したデータベース・オブジェクトの設計

ビューとストアド・プロシージャは、ユーザがアクセスできるデータと実行できるタスクをチューニングする代替方法を提供します。

次の項を参照してください。

- 「プロシージャとトリガの利点」『SQL Anywhere サーバ - SQL の使用法』
- 「高度なセキュリティを実現するためのビューとプロシージャの使い方」 396 ページ

保護された機能の指定

ユーザが使用できるデータベース機能を制限するために、データベース・サーバの起動時に機能保護オプション (-sf) を指定できます。機能保護オプションでは、次のような機能の使用可能性を指定できます。

- サーバ側のバックアップ
- 外部ストアド・プロシージャ
- リモート・データ・アクセス
- Web サービス

機能の完全なリストについては、「-sf サーバ・オプション」 192 ページを参照してください。

データベース・サーバを起動するときには、-sk オプションを指定することもできます。このオプションでは、特定の接続で保護されている機能を有効にするキーを指定します。特定の接続で保護されている機能を有効にするには、secure_feature_key テンポラリ・オプションをデータベース・サーバの起動時に -sk で指定された値に設定します。

sa_server_option システム・プロシージャを使用して、無効になっている機能または機能セットに変更を加えるためには、-sk でキーを指定し、secure_feature_key テンポラリ・オプションをそのキーの値に設定する必要があります。機能を無効または有効にするために行った変更は、直ちに有効になります。

◆ データベース機能を保護するには

1. -sf オプションを指定してデータベース・サーバを起動します。必要に応じて -sk オプションも指定します。

   たとえば、次のコマンドは、データベース・サーバを起動し、リモート・データ・アクセスの使用を無効にします。ただし、無効になっている機能を有効にするためのキーも指定されています。

   ```
   dbsrv10 -n secure_server -sf remote_data_access -sk ls64uwq15 c:¥mydata.db
   ```

2. データベースに接続します。

次に例を示します。

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

3. secure_feature_key テンポラリ・オプションをデータベース・サーバの起動時に -sk で指定された値に設定します。

   次に例を示します。

   ```
   SET TEMPORARY OPTION secure_feature_key = 'ls64uwq15';
   ```

4. sa_server_option システム・プロシージャを使用して、このデータベース・サーバで保護されている機能を変更します。

   次に例を示します。

   ```
   CALL sa_server_option( 'SecureFeatures', '-remote_data_access' );
   ```

参照

- 「-sf サーバ・オプション」 192 ページ
- 「-sk サーバ・オプション」 195 ページ
- 「secure_feature_key [データベース]」 494 ページ
- 「sa_server_option システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』

データベース・アクティビティの監査

各データベースには、関連付けられたトランザクション・ログ・ファイルがあります。トランザクション・ログはデータベースのリカバリに使用します。これは、データベースに対して実行されたトランザクションの記録です。「トランザクション・ログ」 816 ページを参照してください。

トランザクション・ログには、実行されたすべてのデータ定義文と、それを実行したユーザ ID が格納されます。また、すべての更新、削除、挿入、これらの文を実行したユーザも格納されます。ただし、監査の目的によっては、これでは不十分です。デフォルトでは、トランザクション・ログにはイベントの時間は含まれず、イベントが発生した順序だけが含まれます。また、失敗したイベントや、SELECT 文も含まれません。

監査は、データベース上で行われたアクティビティをトラッキングする方法です。監査を使用すると、追加のデータがトランザクション・ログに保存されます。

- すべてのログイン試行 (成否とも)。ターミナル ID を含む。
- すべてのイベントの正確なタイムスタンプ (ミリ秒まで解析)。
- すべてのパーミッションの検査 (成否とも)。パーミッションが検査されたオブジェクトがあれば、それも含む。
- DBA 権限を必要とするすべてのアクション。

データベースで監査が有効になっている場合、トランザクション・ログの使用を停止することはできません。トランザクション・ログをオフにするためには、先に監査をオフにする必要があります。

監査の有効化

データベース管理者が「監査」を有効にすると、セキュリティ関連の情報がトランザクション・ログに追加されます。

監査はデフォルトでは無効になっています。データベース上で監査を有効にするには、DBA でパブリック・オプション auditing の値を On に設定します。この場合、auditing オプションの値を OFF に設定して明示的に無効にするまで、監査は有効なままとなります。このプロシージャの設定には、DBA パーミッションが必要です。

- **監査を有効にするには、次の手順に従います。**

1. DBA としてデータベースに接続します。
2. 次の文を実行します。

```
SET OPTION PUBLIC.auditing = 'On';
```

「auditing オプション [データベース]」 428 ページを参照してください。

個々の接続の監査

データベースの監査を有効にした後で、データベース・ログイン・プロシージャに conn_auditing database テンポラリ・オプションを指定し、接続ごとに監査を有効にすることができます。クライアント・コンピュータの IP アドレスや接続のタイプなどの情報に基づいて監査を有効にできます。

ログイン・プロシージャに conn_auditing オプションを設定しなかった場合、このオプションはデフォルトで On に設定されます。

次の例は、DBA ユーザによる接続を除くすべての接続で監査を有効にするログイン・プロシージャの一部を示しています。

```
DECLARE usr VARCHAR(128)
SELECT CONNECTION_PROPERTY( 'Userid' ) INTO usr;
IF usr != 'DBA' THEN
  SET TEMPORARY OPTION conn_auditing='On'
ELSE
  SET TEMPORARY OPTION conn_auditing='Off'
END IF;
```

詳細については、「login_procedure オプション [データベース]」 460 ページおよび「conn_auditing オプション [データベース]」 435 ページを参照してください。

監査情報の取り出し

ログ変換 (dbtran) ユーティリティを使用して、監査情報を取り出すことができます。このユーティリティは、Sybase Central またはコマンド・プロンプトからアクセスできます。dbtran ユーティリティは指定のトランザクション・ログを使用して、各コマンドを実行したユーザの情報と、すべてのトランザクションを保持する SQL スクリプトを生成します。dbtran は、-g オプションを使用することによって、監査情報を含むより多くのコメントを生成します。-g オプションを指定すると、自動的に以下のオプションが設定されます。

- **-d** 出力を日付順に表示する
- **-t** トリガで生成したオペレーションを出力に含める
- **-a** ロールバック・トランザクションを出力に含める

ログ変換ユーティリティは、稼働中のデータベース・サーバに対して、またはデータベース・ログ・ファイルに対して実行できます。

- **稼働中のデータベース・サーバから監査情報を取り出すには、次の手順に従います。**

1. ユーザ ID に DBA 権限があることを確認します。
2. データベース・サーバの稼働中に、システムのコマンド・プロンプトで次の文を実行します。

   ```
   dbtran -g -c "UID=DBA;PWD=sql;..." -n demo.sql
   ```

   接続文字列の詳細については、「接続パラメータ」 230 ページを参照してください。

♦ **トランザクション・ログ・ファイルから監査情報を取り出すには、次の手順に従います。**

1. データベース・サーバを閉じて、ログ・ファイルが使用可能であることを確認します。
2. システムのコマンド・プロンプトで次の文を実行して、ファイル *demo.log* の情報をファイル *demo.sql* に格納します。

```
dbtran -g demo.log
```

詳細については、「ログ変換ユーティリティ (dbtran)」 688 ページを参照してください。

監査コメントの追加

sa_audit_string システム・ストアド・プロシージャを使用して、監査証跡にコメントを追加できます。これは引数を 1 つとります。引数は 200 バイト以内の文字列です。このプロシージャを呼び出すには、DBA パーミッションが必要です。

次に例を示します。

```
CALL sa_audit_string( 'Started audit testing here.' );
```

このコメントは監査文としてトランザクション・ログに格納されます。

監査の例

この例では、権限のない情報へのアクセス試行を、監査機能がどのようにして記録するかを示します。

1. データベース管理者として、監査を有効にします。

   これは、次のように Sybase Central から実行できます。

   - SQL Anywhere 10 Demo データ・ソースを使用して接続します。このとき DBA ユーザとして接続します。
   - コンテキスト・ドロップダウン・リストから、**demo - DBA** を選択して SQL Anywhere サンプル・データベースを選択し、[ファイル] メニューの [オプション] を選択します。
   - オプションのリストから [Auditing] を選択し、[値] フィールドに値 **On** を入力します。[設定] をクリックしてオプションを設定し、[閉じる] をクリックします。

   または、Interactive SQL を使用することもできます。Interactive SQL から DBA としてサンプル・データベースに接続し、次の文を実行します。

```
SET OPTION PUBLIC.auditing = 'On';
```

2. パスワードが **BadUser** のユーザ BadUser をサンプル・データベースに追加します。これは、Sybase Central から実行できます。または、Interactive SQL を使用して、次の文を入力することもできます。

```
GRANT CONNECT TO BadUser
IDENTIFIED BY 'BadUser';
```

3. Interactive SQL を使用して、BadUser としてサンプル・データベースに接続し、次に示すクエリで **Employees** テーブルの機密情報にアクセスを試みます。

   ```
   SELECT Surname, Salary
   FROM GROUPO.Employees;
   ```

   「**パーミッションがありません： "Employees" から選択するためのパーミッションがありません。**」というエラー・メッセージが表示されます。

4. コマンド・プロンプトで次のコマンドを実行します。

   ```
   dbtran -g -c "DSN=SQL Anywhere 10 Demo" -n demo.sql
   ```

   このコマンドは、トランザクション・ログ情報と、監査情報を持つ一連のコメントを含む *demo.sql* ファイルを生成します。権限のない BadUser による Employees テーブルへのアクセス試行を示す行は、ファイルに次のように含まれています。

   ```
   --AUDIT-1001-0000287812 -- 2004/02/11 13:59:58.765 Checking Select permission on Employees
   - Failed
   --AUDIT-1001-0000287847 -- 2004/02/11 13:59:58.765 Checking Select permission on Employees
   (Salary) - Failed
   ```

5. このマニュアル内で実行するその他の例が予期した結果になるように、サンプル・データベースを元の状態にリストアします。

   DBA ユーザとして接続し、次の操作を実行します。

   - ユーザ ID **BadUser** から接続の権限を取り消します。
   - PUBLIC.auditing オプションを Off に設定します。

データベース・サーバ外のアクションの監査

データベース・ユーティリティの中には、データベース・ファイルに直接作用するものがあります。セキュリティを考慮した環境では、信用のおけるユーザ以外はデータベース・ファイルにアクセスできないようにしてください。

アクションの監査を実行するとき、Windows または UNIX の場合にかぎっては、dbtran または dblog を使用して、データベース・ファイルと同じディレクトリに拡張子 *.alg* のテキスト・ファイルを生成します。たとえば、*demo.db* の場合は、*demo.alg* というファイルが生成されます。ツール名、Windows または UNIX ユーザ名、日付／時刻を含むレコードがこのファイルに追加されます。auditing オプションが On に設定されている場合は、レコードが *.alg* ファイルに追加されるだけです。

参照

- 「ログ変換ユーティリティ (dbtran)」 688 ページ
- 「トランザクション・ログ・ユーティリティ (dblog)」 735 ページ

安全な方法でのデータベース・サーバの実行

データベース・サーバ起動時、またはサーバのオペレーション中に設定できる、次のようなセキュリティ機能があります。

- **データベースの開始と停止**　パーソナル・データベース・サーバを使用している場合、デフォルトでは、すべてのユーザが実行中のサーバ上で追加のデータベースを起動できます。ネットワーク・データベース・サーバの場合、デフォルトでは、実行中のデータベース・サーバ上で別のデータベースを起動するためには DBA 権限が必要です。-gd オプションは、この機能を実行できるユーザを、すでに接続しているデータベースで特定のレベルのパーミッションを付与されているユーザに制限できます。このオプションに指定できる値は、DBA、all、none です。「-gd サーバ・オプション」 168 ページを参照してください。

- **データベースの作成と削除**
  パーソナル・データベース・サーバを実行している場合、デフォルトでは、すべてのユーザが CREATE DATABASE 文を使用してデータベース・ファイルを作成できます。ネットワーク・データベース・サーバの場合、デフォルトではデータベースの作成に DBA 権限が必要です。-gu オプションは、この機能を実行できるユーザを、すでに接続しているデータベースで特定のレベルのパーミッションを付与されているユーザに制限できます。このオプションに指定できる値は、DBA、all、none、utility_db です。「-gu サーバ・オプション」 176 ページを参照してください。

- **サーバの停止**　dbstop ユーティリティは、データベース・サーバを停止します。このユーティリティは、バッチ・ファイルや、サーバを対話形式で停止 (サーバ・メッセージ・ウィンドウの [シャットダウン] をクリックして行う) できない場合に便利です。パーソナル・データベース・サーバの場合、デフォルトでは、すべてのユーザが dbstop を実行してサーバを停止できます。ネットワーク・データベース・サーバの場合、デフォルトではデータベース・サーバの停止に DBA 権限が必要です。-gk オプションは、この機能を実行できるユーザを、データベースで特定のレベルのパーミッションを付与されているユーザに制限できます。このオプションに指定できる値は、DBA、all、none です。「-gk サーバ・オプション」 170 ページを参照してください。

- **データのロードとアンロード**
  LOAD TABLE 文、UNLOAD TABLE 文、UNLOAD 文はすべて、データベース・サーバ・コンピュータ上のファイル・システムにアクセスできます。UNIX 以外のオペレーティング・システムを使用するパーソナル・データベース・サーバの場合、デフォルト設定は all です。ネットワーク・データベース・サーバや UNIX パーソナル・サーバの場合、デフォルト設定は DBA です。パーソナル・データベース・サーバが稼働中の場合、すでにファイル・システムにアクセスできるため、セキュリティ上の問題はありません。ネットワーク・データベース・サーバを稼働中の場合は、ファイル・システムへの不当なアクセスによってセキュリティ問題の起こる可能性があります。-gl オプションを使用して、データのロードとアンロードを行うのに必要なデータベース・パーミッションを制御できます。このオプションに指定できる値は、DBA、all、none です。「-gl サーバ・オプション」 171 ページを参照してください。

- **トランスポート・レイヤ・セキュリティによるクライアント／サーバ通信の暗号化**　トランスポート・レイヤ・セキュリティを使用して、クライアント・アプリケーションとデータベース・サーバ間の通信を認証することで、ネットワーク・パケットのセキュリティを高めることができます。トランスポート・レイヤ・セキュリティでは、楕円曲線暗号方式または RSA 暗号

方式を使用します。「トランスポート・レイヤ・セキュリティ」 949 ページを参照してください。

- **データベース機能の無効化** -sf サーバ・オプションを使用すると、データベース・サーバ上で実行されているデータベースで無効にする機能のリストを指定できます。このリストに含まれる機能は、クライアント・アプリケーションに加え、データベース内に定義されているストアド・プロシージャ、トリガ、イベントでも使用できません。この設定が役に立つのは、使用するデータベースが自分の所有しているものではないため、ウイルスやトロイの木馬などの望ましくないアクションが組み込まれている可能性がある場合です。「-sf サーバ・オプション」 192 ページを参照してください。

データベースの暗号化

データベース管理者として、データベースの暗号化を使用して第三者によるデータの解読を困難にできます。データベースを安全に管理するために、単純暗号化または強力な暗号化のいずれかを選択できます。

単純暗号化

単純暗号化は、難読化と同じです。これにより第三者は、ディスク・ユーティリティを使用してファイルを表示し、データベースのデータを解読することが困難になります。単純暗号化では、データベースの暗号化のためのキーは不要です。単純暗号化方式は、旧バージョンの SQL Anywhere でサポートされています。

♦ **単純暗号化を使用するには、次の手順に従います。**

1. dbinit -e オプションを使用して、データベースを作成します。

   次の例では、単純暗号化を使用して、データベース *test.db* を作成します。

   ```
   dbinit -e test.db
   ```

参照

- 「初期化ユーティリティ (dbinit)」 662 ページ
- 「CREATE DATABASE 文」 『SQL Anywhere サーバ - SQL リファレンス』

強力な暗号化

強力なデータベース暗号化方式では、キー (パスワード) がないとデータベースの操作やアクセスを行うことができません。アルゴリズムは、データベースやトランザクション・ログ・ファイルに含まれる情報をエンコードして解読できないようにしています。

> **警告**
> キーは保護してください。キーのコピーは、安全な場所に保管してください。キーを紛失すると、データベースにまったくアクセスできなくなり、リカバリも不可能になります。

暗号化アルゴリズム

AES は、SQL Anywhere の強力な暗号化を実装するために使用されているアルゴリズムです。これは、米国商務省標準技術局 (NIST：National Institute of Standards and Technology) によってブロック暗号のための新しい次世代標準暗号化方式 (AES：Advanced Encryption Standard) として選択されたブロック暗号化アルゴリズムです。これは、パフォーマンスやサイズの面で SQL Anywhere データベースの暗号化に役立つ多くのプロパティを備えています。

AES_FIPS タイプを使用した強力な暗号化を実装するために、FIPS 承認の AES アルゴリズムを指定することもできます。-fips オプションを指定してデータベース・サーバを起動する場合、AES または AES_FIPS の強力な暗号化方式で暗号化されたデータベースを実行できますが、単純

暗号化方式で暗号化されたデータベースは実行できません。-fips を指定する場合、暗号化されていないデータベースをサーバ上で起動することもできます。「-fips サーバ・オプション」 166 ページを参照してください。

AES_FIPS で暗号化したデータベースを実行するために使用するコンピュータには、SQL Anywhere セキュリティ・オプションをインストールしてください。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

注意
FIPS は、すべてのプラットフォームで使用できるわけではありません。サポートされているプラットフォームのリストについては、「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」の SQL Anywhere、Ultra Light、Mobile Link の各表にある「別途ライセンスが必要なコンポーネント」を参照してください。

強力に暗号化されているデータベースの作成

強力な暗号化を使用して新しいデータベースを作成するには、次の方法を使用できます。

- データベース初期化ユーティリティ (dbinit) と、強力な暗号化を有効にするための各種オプションの組み合わせ。

  dbinit ユーティリティの -ek オプションと -ep オプションを使用すると、強力な暗号化を使用するデータベースが作成され、プロンプト・ボックスまたはコマンド・ラインで暗号化キーを指定できます。dbinit -ea オプションは、暗号化アルゴリズムを AES、または FIPS 認定アルゴリズムの場合は AES_FIPS に設定します。「初期化ユーティリティ (dbinit)」 662 ページを参照してください。

- CREATE DATABASE 文の ENCRYPTION 句。KEY オプションは暗号化キーを設定し、ALGORITHM オプションは暗号化アルゴリズムを AES、または FIPS 認定アルゴリズムの場合は AES_FIPS に設定します。「CREATE DATABASE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  また、Sybase Central の [データベース作成] ウィザードを使用して、強力に暗号化されたデータベースを作成することもできます。

- データベースのアンロード・ユーティリティ (dbunload) と、新規データベースを強力な暗号化で作成するためのオプション。-an オプションは、新規データベースを作成します。強力な暗号化と暗号化キーをプロンプト・ボックスまたはコマンド・ラインで指定するには、-ek オプションまたは -ep オプションを使用します。-ea オプションは、暗号化アルゴリズムを AES、または FIPS 認定アルゴリズムの場合は AES_FIPS に設定します。

また、Sybase Central の [データベースのアンロード] ウィザードを使用して、強力に暗号化されたデータベースを作成することもできます。

詳細については、「[データベース・アンロード] ウィザードの使用」『SQL Anywhere サーバ - SQL の使用法』および「アンロード・ユーティリティ (dbunload)」 739 ページを参照してください。

♦ 強力に暗号化されたデータベースを作成するには、次の手順に従います (SQL の場合)。

1. Interactive SQL から既存のデータベースに接続します。
2. ENCRYPTION 句、KEY オプション、ALGORITHM オプションを含む CREATE DATABASE 文を実行します。

   たとえば、次の文は、FIPS 認定 AES 暗号化を使用して、*c:¥* ディレクトリにデータベース・ファイル *myencrypteddb.db* を作成します。

```
CREATE DATABASE 'c:¥¥myencrypteddb'
TRANSACTION LOG ON
ENCRYPTED ON
  KEY '0kZ2o52AK#'
  ALGORITHM 'AES_FIPS';
```

♦ 強力に暗号化されたデータベースを作成するには、次の手順に従います (コマンド・プロンプトの場合)。

1. コマンド・プロンプトで、dbinit ユーティリティを使用してデータベースを作成します。コマンド・プロンプトまたはダイアログ・ボックスで暗号化キーを指定するには、-ek または -ep をそれぞれ指定します。

   次のコマンドは、強力に暗号化されたデータベースを作成し、暗号化キーとアルゴリズムを指定します。

```
dbinit -ek "0kZ2o56AK#" -ea AES_FIPS "myencrypteddb.db"
```

2. コマンド・プロンプトからデータベースを起動します。

```
dbeng10 myencrypteddb.db -ek "0kZ2o56AK#"
```

暗号化キーの詳細については、「DatabaseKey 接続パラメータ [DBKEY]」 242 ページを参照してください。

暗号化が必要なデータベースがある場合は、CREATE ENCRYPTED FILE 文を使用して暗号化できます。暗号化では、ファイルに上書きするのではなく、ファイルのコピーを暗号化形式で作成します。

注意
テーブル暗号化が有効になっている場合は、データベースを暗号化することはできません。テーブル暗号化を無効にしてデータベースを再作成する必要があります。

♦ **作成済みのデータベースを暗号化するには**

1. CREATE ENCRYPTED FILE 文を使用して、暗号化されていないデータベースを暗号化します。

   次の例では、データベース・ファイル *current.db* の暗号化されたコピーを作成し、そのコピーに *encrypted.db* という名前を付けます。

   ```
   CREATE ENCRYPTED FILE encrypted.db
   FROM current.db
   KEY abc
   ALGORITHM AES;
   ```

2. 同じキー情報を使用し、データベース・ファイルの命名規則に従って、このデータベースのトランザクション・ログ・ファイル、DB 領域ファイル、ミラー・ログ・ファイル (ある場合) を CREATE ENCRYPTED FILE 文によって暗号化します。「CREATE ENCRYPTED FILE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

注意
CREATE ENCRYPTED FILE 文では、暗号化されていない形式で作成されたデータベースを暗号化することはできますが、暗号化が有効になっていないデータベースのテーブル暗号化だけを有効にすることはできません。データベースの暗号化を有効にするためには、データベースを再作成し、テーブル暗号化を有効にする必要があります。「テーブル暗号化を有効にする」 944 ページを参照してください。

CREATE DECRYPTED FILE 文を使用して、データベースを復号化することができます。CREATE ENCRYPTED FILE 文の場合と同じように、ファイルに上書きするのではなく、ファイルのコピー (この場合は復号化形式) を作成します。データベース・ファイルだけでなく、関連するトランザクション・ログ・ファイルと DB 領域ファイルも復号化する必要があります。「CREATE DECRYPTED FILE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

暗号化キーの使用

ほとんどのパスワードと同様、最善の方法は、簡単には推測できないキー値を選択することです。キーには 8 ー 30 文字の値を選択し、大文字と小文字、数字、文字、特殊文字を組み合わせて使用することをおすすめします。

警告
キーのコピーは、安全な場所に保管してください。キーは、データベースを起動したり変更したりするたびに必要になります。キーを紛失すると、データベースにまったくアクセスできなくなり、リカバリも不可能になります。

CREATE ENCRYPTED FILE を使用して、暗号化されたデータベースやテーブル暗号化が有効になっているデータベースの暗号化キーを変更することができます。データベースを暗号化する場合と同じように、既存のファイルに上書きするのではなく、ファイルのコピーを作成し、そのコピーを新しいキーで暗号化します。

♦ **データベースの暗号化キーを変更するには、次の手順に従います。**

1. CREATE ENCRYPTED FILE 文を使用して、暗号化されたデータベースの暗号化キーを変更します。

   次の例では、abc というキーで暗号化されたデータベース・ファイル *currentkey.db* のコピーを作成し、そのコピーに *newkey.db* という名前を付けて、暗号化キー abc123 で暗号化します。

   ```
   CREATE ENCRYPTED FILE newkey.db
   FROM currentkey.db
   KEY abc123
   OLD KEY abc
   ALGORITHM AES;
   ```

2. 同じキー情報を使用し、データベース・ファイルの命名規則に従って、このデータベースのトランザクション・ログ・ファイル、DB 領域ファイル、ミラー・ログ・ファイル (ある場合) を CREATE ENCRYPTED FILE 文によって暗号化します。「CREATE ENCRYPTED FILE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

暗号化キーの選択

ほとんどのパスワードと同様、最善の方法は、簡単には推測できないキー値を選択することです。キーの長さは任意ですが、短いと推測されやすいため、一般的には長い方が適しています。また、数字、文字、特殊文字を組み合わせると、キーは推測されにくくなります。

データベースを起動するたびに、キーを指定してください。キーを忘れた場合はデータベースにまったくアクセスできなくなります。

暗号化キーの保護

暗号化キーの入力に、コマンド・プロンプト (デフォルト) またはプロンプト・ボックスのいずれかを選択できます。プロンプト・ボックスでのキー入力を選択すると、キーが表示されないため、さらにセキュリティが強化されます。クライアントでは、データベースを起動するたびにキーを指定してください。データベース管理者がデータベースを起動する場合は、クライアントでキーを使用する必要はありません。「-ep サーバ・オプション」 164 ページを参照してください。

強力な暗号化の制御

SQL Anywhere では、データベース管理者が管理する強力な暗号化のテクノロジは 4 つあります。それは、強力な暗号化のステータス、暗号化キー、暗号化キーの保護、暗号化アルゴリズムです。

強力な暗号化のステータス

既存のデータベースでは強力な暗号化のオンとオフを簡単に切り替えることはできませんが、強力な暗号化の実装は、2 つのオプションから選択できます。強力な暗号化を指定して新規にデー

タベースを作成することもできれば、既存のデータベースを再構築して同時に暗号化ステータスを変更することもでき、さらには既存のデータベースに対して CREATE ENCRYPTED FILE 文を実行することもできます。

データベースを再構築して、既存のデータベースに含まれるすべてのデータとスキーマをアンロードできます。新しいデータベースを作成して (ここで強力な暗号化のステータスを含めたさまざまな設定を変更できます)、データを新しいデータベースに再ロードします。強力に暗号化されたデータベースをアンロードするにはキーが必要です。

参照

- 「データベースの再ロード」 『SQL Anywhere サーバ - SQL の使用法』
- 「CREATE DATABASE 文」 『SQL Anywhere サーバ - SQL リファレンス』

パフォーマンスの問題

データベースを暗号化している場合、SQL Anywhere のパフォーマンスが多少低下します。パフォーマンスの影響は、ディスクとのページの読み取りや書き込みの頻度によって異なります。また、サーバが使用するキャッシュ・サイズを適切に設定することによって影響を最小限にできます。

キャッシュの初期サイズを増やすには、サーバの起動時に -c オプションで指定します。キャッシュの動的なサイズ変更がサポートされているオペレーティング・システムでは、使用されるキャッシュ・サイズが、使用可能なメモリの容量によって制限される場合があります。そのため、キャッシュ・サイズを増加するには、使用可能なメモリを増加します。

参照

- 「パフォーマンス向上へのキャッシュの使用」 『SQL Anywhere サーバ - SQL の使用法』
- 「-c サーバ・オプション」 149 ページ

データベースの一部の暗号化

データベースの一部のみを暗号化する場合は、ENCRYPT 関数を使用します。ENCRYPT 関数は、同じ AES の強力な暗号化アルゴリズムを使用します。このアルゴリズムはデータベースの暗号化用に使用され、その関数に渡される値を暗号化します。

ENCRYPT 関数のキーは、大文字と小文字を区別しないデータベース内であっても、大文字と小文字が区別されます。ほとんどのパスワードと同様、最善の方法は、簡単には推測できないキー値を選択することです。キーには最低でも 16 文字の値を選択し、大文字と小文字、数字、文字、特殊文字を組み合わせて使用することをおすすめします。このキーは、データを復号化するたびに必要になります。

警告
キーは保護してください。キーのコピーは、安全な場所に保管してください。キーを紛失すると、暗号化データにまったくアクセスできなくなり、そこからのリカバリも不可能になります。

暗号化された値は、DECRYPT 関数で復号化できます。このとき、ENCRYPT 関数で指定したキーと同じキーを使用する必要があります。これらの関数はともに LONG BINARY 値を返します。異なるデータ型を使用する必要がある場合は、CAST 関数を使用して、その値を必要なデータ型に変換できます。次の例では、CAST 関数を使用して、復号化された値を必要なデータ型に変換する方法を示します。「CAST 関数 [データ型変換]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

データベース・ユーザが復号化された形式のデータにアクセスする必要があっても、暗号化キーにはアクセスできないようにする必要がある場合は、DECRYPT 関数を使用するビューを作成できます。これにより、ユーザは暗号化キーを知らなくても、復号化されたデータにアクセスできるようになります。テーブルを使用したビューまたはストアド・プロシージャを作成する場合は、ALTER VIEW 文や ALTER PROCEDURES 文の SET HIDDEN パラメータを使用して、ユーザがビュー定義やプロシージャ定義を参照することによって暗号化キーにアクセスできないようにすることができます。「ALTER PROCEDURE 文」 『SQL Anywhere サーバ - SQL リファレンス』と「ALTER VIEW 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

カラムの暗号化の例

次の例では、user_info というテーブルのパスワードを格納するカラムを暗号化するトリガを使用します。user_info テーブルは、次のように定義されています。

```
CREATE TABLE user_info (
  employee_ID INTEGER NOT NULL PRIMARY KEY,
  user_name CHAR(80),
  user_pwd CHAR(80) );
```

新しいユーザが追加されたとき、または既存のユーザのパスワードが更新されたときに、2 つのトリガが user_pwd カラムの値を暗号化するためにデータベースに追加されます。

- encrypt_new_user_pwd トリガは、新しいローが user_info テーブルに追加されるたびに実行されます。

  ```
  CREATE TRIGGER encrypt_new_user_pwd
  BEFORE INSERT
  ON user_info
  REFERENCING NEW AS new_pwd
  FOR EACH ROW
  BEGIN
     SET new_pwd.user_pwd=ENCRYPT(new_pwd.user_pwd, '8U3dkA');
  END;
  ```

- encrypt_updated_pwd トリガは、user_info テーブルの user_pwd カラムが更新されるたびに実行されます。

  ```
  CREATE TRIGGER encrypt_updated_pwd
  BEFORE UPDATE OF user_pwd
  ON user_info
  REFERENCING NEW AS new_pwd
  FOR EACH ROW
  BEGIN
     SET new_pwd.user_pwd=ENCRYPT(new_pwd.user_pwd, '8U3dkA');
  END;
  ```

データベースに新しいユーザを追加する場合

```
INSERT INTO user_info
VALUES ( '1', 'd_williamson', 'abc123');
```


SELECT 文を発行して user_info テーブルの情報を表示する場合、user_pwd カラムの値はバイナリ・データ (パスワードの暗号化された形式) であり、INSERT 文で指定された値 abc123 ではありません。

このユーザがパスワードを変更した場合

```
UPDATE user_info
SET user_pwd='xyz'
WHERE employee_ID='1';
```

encrypt_updated_pwd トリガが実行され、新しいパスワードの暗号化された形式が user_pwd カラムに表示されます。

元のパスワードは、次の SQL 文を発行して検索できます。この文はデータを復号化するために DECRYPT 関数と暗号化キーを使用し、値を LONG BINARY から CHAR 値に変換するために CAST 関数を使用しています。

```
SELECT CAST (DECRYPT(user-pwd, '8U3dkA') AS CHAR(100)) FROM user_info
WHERE employee_ID = '1';
```

参照

- 「ENCRYPT 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』
- 「DECRYPT 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』

テーブル暗号化

テーブル暗号化によって、データベース全体の暗号化がもたらすようなパフォーマンスの低下を招くことなく、機密データが含まれるテーブルや実体化されたビューを暗号化することができます。テーブルの暗号化が有効な場合、暗号化されたテーブルのテーブル・ページ、関連するインデックス・ページ、テンポラリ・ファイルのページが暗号化されます。さらに、暗号化されたテーブルのトランザクションを含むトランザクション・ログ・ページも暗号化されます。

実体化されたビューの暗号化については、「実体化ビュー (Materialized View) の暗号化と復号化」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

データベース内のテーブルを暗号化するためには、テーブル暗号化を有効にしておく必要があります。テーブル暗号化の有効化は、データベースを初期化するときに行います。テーブル暗号化が有効になっているかどうかを確認するには、次のように DB_PROPERTY 関数を使用して EncryptionScope データベース・プロパティの値を取得します。

```
SELECT DB_PROPERTY( 'EncryptionScope' );
```

TABLE が返された場合は、テーブル暗号化が有効になっています。

テーブル暗号化で暗号化アルゴリズムが有効であるかどうかを確認するには、次のように DB_PROPERTY 関数を使用して Encryption データベース・プロパティの値を取得します。

```
SELECT DB_PROPERTY( 'Encryption' );
```

サポートされている暗号化アルゴリズムのリストについては、「データベースの暗号化」 936 ページを参照してください。

テーブル暗号化がパフォーマンスに及ぼす影響

暗号化されたテーブルでは、各テーブル・ページがディスクへの書き込みと同時に暗号化され、ディスクから読み取るときに復号化されます。このプロセスはアプリケーションには影響しません。ただし、暗号化されたテーブルの読み込みや書き込みにおいてパフォーマンスが多少低下することがあります。既存のテーブルを暗号化または復号化する場合、テーブルのサイズによっては時間がかかることがあります。

暗号化されたテーブル内のカラムがインデックス付けされている場合、データベースのテンポラリ・ファイルと同じように、暗号化されたテーブルのトランザクションを含むトランザクション・ログ・ページも暗号化されます。その他のデータベースとトランザクション・ログ・ページは暗号化されません。

暗号化されたテーブルに圧縮されたカラムが含まれている場合があります。その場合、データは圧縮されてから暗号化されます。

テーブルの暗号化は必要記憶域には影響しません。

テーブル暗号化が有効であるデータベースの起動

テーブル暗号化が有効であるデータベースを起動する方法は、暗号化されたデータベースを起動する場合と同じです。たとえば、-ek オプションを指定してデータベースを起動する場合は、キーを指定する必要があります。-ep オプションを指定してデータベースを起動すると、キーの入力を要求されます。「初期化ユーティリティ (dbinit)」 662 ページを参照してください。

テーブル暗号化を有効にする

テーブル暗号化を有効にする場合は、データベースの作成時に行う必要があります。データベースでテーブル暗号化が有効になっていない場合、またはデータベース暗号化が有効な場合は、テーブル暗号化を有効にしてデータベースを再作成する必要があります。

♦ 単純なテーブル暗号化を有効にしてデータベースを作成するには (SQL)

- CREATE DATABASE 文を使用してデータベースを作成します。ただし、キーやアルゴリズムを設定するオプションは指定しません。

  次のコマンドは、テーブルの単純暗号化を有効にしてデータベース *new.db* を作成します。

  ```
  CREATE DATABASE new.db ENCRYPTED TABLE;
  ```

  このデータベース内のテーブルを暗号化するときには、単純暗号化アルゴリズムが使用されます。

♦ 強力なテーブル暗号化を有効にしてデータベースを作成するには (SQL)

- CREATE DATABASE 文を使用してデータベースを作成します。このとき、キーと暗号化アルゴリズムを指定します。

  次のコマンドは、キー abc と暗号化アルゴリズム AES_FIPS を使用する強力なテーブル暗号化を有効にしてデータベース *new.db* を作成します。

  ```
  CREATE DATABASE new.db
  ENCRYPTED TABLE
  ```

```
KEY abc
ALGORITHM AES_FIPS;
```

このデータベース内のテーブルを暗号化するときには、キー abc とともに AES_FIPS アルゴリズムが使用されます。

♦ 単純なテーブル暗号化を有効にしてデータベースを作成するには (コマンド・プロンプト)

1. dbinit に -et オプションを指定してデータベースを作成します。キーと暗号化アルゴリズムは指定しません。

   次のコマンドは、テーブルの単純暗号化を有効にしてデータベース *new.db* を作成します。

   ```
   dbinit new.db -et
   ```

   このデータベース内のテーブルを暗号化するときには、単純暗号化アルゴリズムが使用されます。

♦ 強力なテーブル暗号化を有効にしてデータベースを作成するには (コマンド・プロンプト)

1. dbinit に -et オプションと -ek オプションを指定し、キーと暗号化アルゴリズムも指定して、データベースを作成します。

   次のコマンドは、キー abc と暗号化アルゴリズム AES_FIPS を使用する強力なテーブル暗号化を有効にしてデータベース *new.db* を作成します。

   ```
   dbinit new.db -et -ek abc -ea AES_FIPS
   ```

   このデータベース内のテーブルを暗号化するときには、キー abc とともに AES_FIPS アルゴリズムが使用されます。

参照

- 「初期化ユーティリティ (dbinit)」 662 ページ
- 「CREATE DATABASE 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「データベースの作成 (Sybase Central の場合)」 『SQL Anywhere サーバ - SQL の使用法』

テーブルの暗号化

データベース内のテーブルを暗号化するためには、そのデータベースでテーブル暗号化が有効になっている必要があります。

テーブルを暗号化するときには、データベースの作成時に指定されたテーブル暗号化設定 (単純暗号化、AES、FIPS など) がテーブルに適用されます。

♦ テーブルの作成時にテーブルを暗号化するには

1. CREATE TABLE 文の ENCRYPTED 句を使用してテーブルを作成します。

   次のコマンドは、暗号化されたテーブル Employees を作成します。

   ```
   CREATE TABLE Employees (
    MemberID CHAR(40),
   ```



```
 CardNumber INTEGER )
ENCRYPTED;
```

♦ 作成済みのテーブルを復号化するには

- ALTER TABLE 文の NOT ENCRYPTED 句を使用してテーブルを復号化します。

  次のコマンドは、Employees テーブルを復号化します。

  ```
  ALTER TABLE Employees
  NOT ENCRYPTED;
  ```

参照

- 「データベースの暗号化」 936 ページ
- 「テーブル暗号化を有効にする」 944 ページ
- 「CREATE TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』
- 「ALTER TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』

Windows CE データベースの保護

ここでは、Windows CE データベースの安全管理に役立つ SQL Anywhere の機能について説明します。特に、監査とデータベース暗号化について説明します。他のセキュリティ機能の概要についても説明し、詳細情報の参照先も示しています。

データベース・ファイルの暗号化や単純な通信暗号化など、Windows デスクトップ・プラットフォームを対象とする SQL Anywhere セキュリティ機能の多くは、Windows CE でもサポートされています。または、ログ変換ユーティリティのように、サポートが変更されているものもあります。

Windows CE 上で動作するデータベースは、Windows デスクトップ・プラットフォームで動作するデータベースと同じユーザ識別情報と認証機能を使用します。これらの機能により、データベースにアクセスできるユーザと、そのユーザが実行できるアクションが制御されます。「データベース・アクセスの制御」 922 ページを参照してください。

Windows CE デバイス・セキュリティ

Windows CE デバイスに機密データを保存する場合は、Windows CE デバイス用に提供されているセキュリティ機能を使用できます。

使用できるセキュリティ機能の詳細については、Windows CE デバイスに付属しているユーザーズ・マニュアルを参照してください。

データベース・サーバ・オプション

サーバ・オプションを使用すると、サーバ上で特定の操作を実行できるユーザを制御できます。

このオプションは、Windows CE デバイス上でデータベースを起動するときに、[サーバ起動オプション] ダイアログの [オプション] フィールドで設定します。

詳細については、「コマンド・ラインからパーミッションを制御する」 22 ページを参照してください。

Windows CE 上でのオプションの設定については、「Windows CE でのサーバ・オプションの指定」 1056 ページを参照してください。

監査

この機能は、トランザクション・ログを使用して、データベース上でのアクションの詳細なレコードを管理します。

監査情報を含めて、トランザクション・ログに保存されている情報を変換するには、ログ変換ユーティリティ (dbtran) を使用します。Windows CE では dbtran ユーティリティがサポートされないため、Windows CE デバイスで保存されるログを変換することはできません。このユーティリティを使用するには、トランザクション・ログ・ファイルを PC にコピーします。

詳細については、「データベース・アクティビティの監査」 930 ページを参照してください。

Windows CE 上でのデータベースの暗号化

データベース暗号化機能を使用する際の、暗号化のレベルを選択します。データベースを安全に管理するために、単純暗号化または強力な暗号化のいずれかを選択できます。SQL Anywhere は、Windows CE 上で、単純暗号化と強力な暗号化の両方をサポートしています。

単純暗号化 このレベルの暗号化は、難読化と同じです。これにより第三者は、ディスク・ユーティリティを使用してファイルを表示し、データベースのデータを解読することが困難になります。単純暗号化では、データベースの暗号化のためのキーは不要です。

単純暗号化方式は、旧バージョンの SQL Anywhere でサポートされています。

強力な暗号化 このレベルの暗号化は、データベースやトランザクション・ログ・ファイルに含まれる情報にスクランブルをかけることで、ディスク・ユーティリティを使用してファイルを表示するだけではデータを解読できないようにします。強力な暗号化にすると、キーなしではデータベースにまったくアクセスできなくなります。Windows CE 上で使用するデータベースを暗号化する場合は、AES アルゴリズムで暗号化してください。

詳細については、「データベースの暗号化」 936 ページを参照してください。

通信の暗号化と Windows CE

クライアント／サーバ通信を暗号化して、ネットワーク上の通信のセキュリティを強化できます。SQL Anywhere は、単純暗号化と強力な暗号化の 2 種類の通信暗号化を備えています。

単純な通信暗号化は、単純暗号化を受けている通信パケットを受け取ります。このレベルの通信暗号化は、Windows CE と以前のバージョンの SQL Anywhere も含め、すべてのプラットフォームでサポートされます。

強力な通信暗号化は、Windows CE では利用できません。

通信の暗号化の詳細については、「Encryption 接続パラメータ [ENC]」 247 ページを参照してください。

第 24 章

トランスポート・レイヤ・セキュリティ

目次

トランスポート・レイヤ・セキュリティの概要

トランスポート・レイヤ・セキュリティは IETF 標準プロトコルであり、デジタル証明書とパブリック・キー暗号方式を使用して、クライアント／サーバ通信をセキュリティ保護します。トランスポート・レイヤ・セキュリティにより、暗号化、改ざん検出、証明書ベースの認証が実現します。

トランスポート・レイヤ・セキュリティは、次の場合に使用できます。

- SQL Anywhere データベース・サーバとクライアント・アプリケーションとの間の通信のセキュリティ保護。
- Mobile Link サーバと Mobile Link クライアントとの間の通信のセキュリティ保護。
- セキュアな SQL Anywhere Web サーバの設定。

セキュリティ保護された通信は、次のメッセージの交換 (ハンドシェイク) で始まります。

- **サーバ認証**　トランスポート・レイヤ・セキュリティでは、セキュア接続の確立と保護にサーバ証明書を使用します。サーバごとにユニークな証明書ファイルを作成します。SQL Anywhere クライアント／サーバ通信用または Mobile Link 同期用のサーバ認証を使用できます。
  - SQL Anywhere クライアント／サーバ通信では、データベース・クライアントは SQL Anywhere データベース・サーバの ID を検証します。
  - Mobile Link 同期では、Mobile Link クライアント (SQL Anywhere または Ultra Light) は Mobile Link サーバの ID を検証します。

効率

トランスポート・レイヤ・セキュリティ・プロトコルでは、パブリック・キー暗号化と対称キー暗号化を組み合わせて使用します。パブリック・キー暗号化は、認証テクニックとして優れていますが、処理量が多くなります。セキュア接続が確立されると、クライアントとサーバはそれ以降の通信に、キー・サイズが 128 ビットで高効率の対称暗号を使用します。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

データベース・ファイルの暗号化の詳細については、次を参照してください。

- SQL Anywhere データベース：「データベースの暗号化」 936 ページ
- Ultra Light データベース：「Ultra Light でのセキュリティの考慮事項」 『Ultra Light - データベース管理とリファレンス』

TLS サポート

RSA、ECC、FIPS の各暗号化は、すべてのプラットフォームで使用できるわけではありません。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

RSA 暗号化

RSA 暗号化は SQL Anywhere では無料で提供されており、クライアント／サーバ通信、同期、Web サービスに使用できます。この実装は FIPS 認定を受けていません。FIPS 認定の RSA 実装には別のライセンスが必要です。

RSA がサポートされているプラットフォームのリストについては、次の項を参照してください。

- RSA 暗号化されたクライアント／サーバ通信：「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」にある SQL Anywhere の表
- RSA 暗号化された Ultra Light 通信：「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」にある Ultra Light の表
- RSA 暗号化された Mobile Link 通信：「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」にある Mobile Link の表

ECC 暗号化

ECC がサポートされているプラットフォームのリストについては、次の項を参照してください。

- ECC 暗号化されたクライアント／サーバ通信：「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」の SQL Anywhere の表にある「別途ライセンスが必要なコンポーネント」
- ECC 暗号化された Ultra Light 通信：「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」の Ultra Light の表にある「別途ライセンスが必要なコンポーネント」
- ECC 暗号化された Mobile Link 通信：「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」の Mobile Link の表にある「別途ライセンスが必要なコンポーネント」

FIPS 承認の暗号化テクノロジ

FIPS 承認のセキュリティ・アルゴリズムを使用すると、データベース・ファイルを暗号化したり、データベース・クライアント／サーバ通信、Web サービス、Mobile Link クライアント／サーバ通信における通信を暗号化できます。連邦情報処理規格 (FIPS) 140-2 では、セキュリティ・アルゴリズムの要件を指定しています。FIPS 140-2 は、米国商務省標準技術局 (NIST：National Institute of Standards and Technology) およびカナダ通信安全保障局 (CSE：Canadian Communication Security Establishment) を通じて、米国政府とカナダ政府から付与されます。

FIPS テクノロジには別のライセンスが必要です。「別途ライセンスが必要なコンポーネント」『SQL Anywhere 10 - 紹介』を参照してください。

SQL Anywhere では、2 つの FIPS 承認モジュールが使用されており、どちらも Certicom 製です。Palm OS の場合、SQL Anywhere では Certicom Security Builder Government Services Edition v1.0.1

を使用します。これは、http://csrc.nist.gov/cryptval/140-1/140val-all.htm ページの 316 番です。Windows (デスクトップと CE) と UNIX プラットフォームの場合、SQL Anywhere は Certicom Security Builder FIPS Module v2.0 を使用します。これは同ページの 542 番です。

トランスポート・レイヤ・セキュリティでは、FIPS を利用できるのは RSA 暗号化だけです (ECC は FIPS プログラムで未対応)。

必要に応じて、FIPS オプションを使用して FIPS の使用を強制できます。FIPS オプションをオンに設定すると、セキュリティ保護された通信はすべて FIPS 承認されたチャネルを経由する必要があります。非 FIPS の RSA を選択した場合は、自動的に FIPS RSA にアップグレードされます。ECC を選択した場合は、エラーがレポートされます。FIPS オプションは、FIPS を強制するコンピュータごとに設定する必要があります。SQL Anywhere サーバと Mobile Link サーバには -fips コマンド・ライン・オプションが用意されています。クライアントには fips オプションがあり、暗号化パラメータを使用して設定できます。

FIPS がサポートされているプラットフォームのリストについては、「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」に記載されている、SQL Anywhere、Ultra Light、Mobile Link の別途ライセンスが必要なコンポーネントの表を参照してください。

SQL Anywhere データベース・ファイルの FIPS テクノロジを使用した暗号化の詳細については、「強力な暗号化」 936 ページを参照してください。

証明書

SQL Anywhere には createcert というツールが付属しています。このツールを使用すると、トランスポート・レイヤ・セキュリティ用の X.509 証明書ファイルを作成できます。一方、サード・パーティ証明書の存在を確認する必要がある場合、またはセキュリティのより高い証明書が必要な場合は、証明書を認証局から購入してください。

トランスポート・レイヤ・セキュリティの設定

次の手順には、トランスポート・レイヤ・セキュリティの設定に必要なタスクの概要が記載されています。

♦ **トランスポート・レイヤ・セキュリティの設定の概要**

1. デジタル証明書を取得します。

   ID ファイルと証明書ファイルが必要です。サーバ ID ファイルにはサーバのプライベート・キーが含まれているので、データベース・サーバまたは Mobile Link サーバにセキュリティ保護された状態で格納する必要があります。サーバの証明書ファイルはクライアントに配布します。

   証明書は認証局から購入することができます。SQL Anywhere には、証明書を作成する機能もあり、特に開発やテストのときに便利です。「デジタル証明書の作成」 955 ページを参照してください。

2. SQL Anywhere クライアント／サーバ・アプリケーション用のトランスポート・レイヤ・セキュリティを設定する場合は、次の手順に従います。

   - **トランスポート・レイヤ・セキュリティを指定して SQL Anywhere データベース・サーバを起動する** -ec データベース・サーバ・オプションを使用して、セキュリティのタイプ、サーバ ID ファイル名、サーバのプライベート・キーを保護するパスワードを指定します。

     「トランスポート・レイヤ・セキュリティを使用するデータベース・サーバの起動」 962 ページを参照してください。

   - **トランスポート・レイヤ・セキュリティを使用するようにクライアント・アプリケーションを設定する** Encryption 接続パラメータ [ENC] を使用して、信頼できる証明書のパスとファイル名を指定します。

     「トランスポート・レイヤ・セキュリティを使用するクライアント・アプリケーションの設定」 963 ページを参照してください。

3. SQL Anywhere Web サービス用のトランスポート・レイヤ・セキュリティを設定する場合は、次の手順に従います。

   - **トランスポート・レイヤ・セキュリティを指定して SQL Anywhere データベース・サーバを起動する** -xs データベース・サーバ・オプションを使用して、セキュリティのタイプ、サーバ ID ファイル名、サーバのプライベート・キーを保護するパスワードを指定します。

   - **ブラウザまたは他の Web クライアントが証明書を信用するように設定する** 「SQL Anywhere Web サービスでのトランスポート・レイヤ・セキュリティの使用」 966 ページを参照してください。

4. Mobile Link 同期用のトランスポート・レイヤ・セキュリティを設定する場合は、次の手順に従います。

- **トランスポート・レイヤ・セキュリティを指定して Mobile Link サーバを起動する** - mlsrv10 -x オプションを使用して、セキュリティ・ストリーム、サーバ ID ファイル名、サーバのプライベート・キーを保護するパスワードを指定します。

  「トランスポート・レイヤ・セキュリティを使用する Mobile Link サーバの起動」 968 ページを参照してください。

- **トランスポート・レイヤ・セキュリティを使用するように Mobile Link クライアントを設定する** Mobile Link 同期クライアント・ユーティリティ (dbmlsync) または Ultra Light アプリケーションを使用して、適切なセキュリティまたはネットワーク・プロトコル・オプションを指定します。セキュリティ・ストリームと信頼されたサーバ証明書ファイル名を指定します。

  「トランスポート・レイヤ・セキュリティを使用する Mobile Link クライアントの設定」 969 ページを参照してください。

デジタル証明書の作成

トランスポート・レイヤ・セキュリティを設定するには、デジタル証明書が必要です。証明書は、認証局から取得することも、SQL Anywhere の機能を使用して作成することもできます。

SQL Anywhere 証明書作成ユーティリティ

SQL Anywhere 証明書作成ユーティリティ createcert は、RSA または ECC を使用して X.509 証明書ファイルを生成します。「証明書作成ユーティリティ [createcert]」 975 ページを参照してください。

SQL Anywhere 証明書ビューワ・ユーティリティ

SQL Anywhere 証明書ビューワ・ユーティリティ viewcert は、RSA または ECC を使用して X.509 証明書を読み込みます。「証明書ビューワ・ユーティリティ [viewcert]」 978 ページを参照してください。

サーバ認証に使用する証明書

サーバ認証に使用する証明書ファイルは、同じ手順で作成できます。どちらの場合も、ID ファイルと証明書ファイルを作成します。

サーバ認証の場合は、サーバ ID ファイルとクライアントに配布する証明書ファイルを作成します。

証明書設定

証明書には、自己署名されたものと、民間またはエンタープライズ認証局によって署名されたものがあります。

- **自己署名証明書**　自己署名されたサーバ証明書は、単純な設定で使用します。「自己署名ルート証明書」 956 ページを参照してください。
- **エンタープライズ・ルート証明書**　エンタープライズ・ルート証明書を使用すると、サーバ証明書に署名できるため、複数のサーバが配備されている環境でのデータの整合性と拡張性が向上します。

  - サーバ証明書の署名に使用するプライベート・キーは、安全な中央のロケーションに保存できます。
  - サーバ認証では、クライアントを再設定しなくても Mobile Link サーバまたはデータベース・サーバを追加できます。

  「証明書チェーン」 956 ページを参照してください。
- **民間認証局**　エンタープライズ・ルート証明書の代わりに、サードパーティの認証局を使用できます。民間認証局は、プライベート・キーを保存するための専用の設備を備えており、高品質なサーバ証明書を作成します。

  詳細については、「証明書チェーン」 956 ページおよび「グローバル署名証明書」 958 ページを参照してください。

自己署名ルート証明書

自己署名ルート証明書は、単一の Mobile Link サーバまたはデータベース・サーバで構成される単純な設定において使用されます。

ヒント
サーバ ID ファイルが複数必要な場合は、エンタープライズ・レベルの証明書チェーンまたは民間認証局を使用します。認証局にはルート・プライベート・キーを格納する専用の設備があり、拡張性と証明書の高度な整合性を提供します。
証明書チェーンの設定については、「証明書チェーン」 956 ページを参照してください。

- **証明書** サーバ認証証明書の場合、自己署名証明書はクライアントに配布されます。自己署名証明書は、識別情報、サーバのパブリック・キー、自己署名されたデジタル署名を含む電子文書です。
- **ID ファイル** サーバ認証証明書の場合、ID ファイルはセキュリティ保護された状態で Mobile Link サーバまたはデータベース・サーバに格納されます。ID ファイルは、自己署名証明書 (クライアントに配布) と、対応するプライベート・キーを組み合わせたものです。プライベート・キーがあると、Mobile Link サーバまたはデータベース・サーバは、初期ハンドシェイクでクライアントから送信されたメッセージを復号できます。

参照

- 「サーバ認証」 969 ページ
- 「トランスポート・レイヤ・セキュリティを使用するデータベース・サーバの起動」 962 ページ
- 「証明書作成ユーティリティ [createcert]」 975 ページ

証明書チェーン

ID ファイルが複数必要な場合、自己署名証明書の代わりに証明書チェーンを使用することで、セキュリティと拡張性を向上させることができます。証明書チェーンでは、ID の署名に認証局またはエンタープライズ・ルート証明書が必要です。

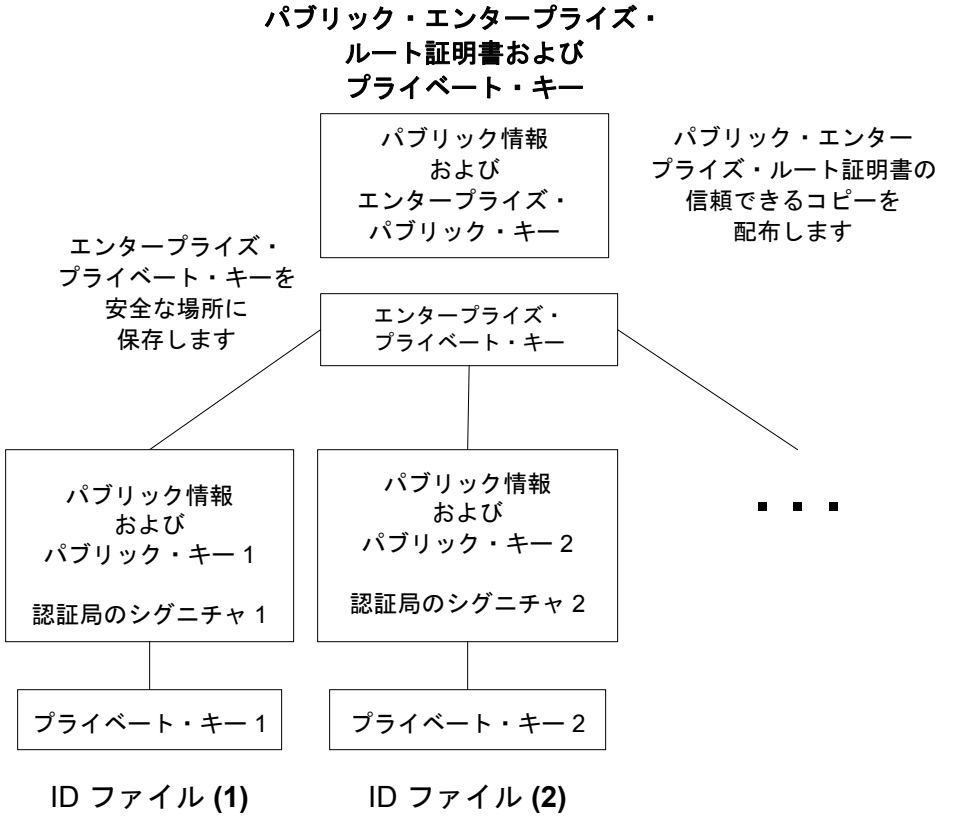
「自己署名ルート証明書」 956 ページを参照してください。

証明書チェーンを使用する利点

証明書チェーンには、次の利点があります。

- **拡張性** サーバ認証の場合、エンタープライズ・ルート証明書または認証局によって署名されたすべての証明書を信頼するようにクライアントを設定できます。新しい Mobile Link サーバまたはデータベース・サーバを追加する場合、クライアントに新しい証明書のコピーは不要です。
- **セキュリティ** エンタープライズ・ルート証明書のプライベート・キーは、ID ファイルにはありません。ルート証明書のプライベート・キーを高セキュリティのロケーションに保存したり、専用の設備を備えている認証局を使用することで、サーバ認証の整合性が保護されます。

次の図は、エンタープライズ・ルート証明書の基本アーキテクチャを示しています。



マルチサーバ環境で使用する証明書を作成するには、次の手順に従います。

- パブリック・エンタープライズ・ルート証明書およびエンタープライズ・プライベート・キーを生成します。

  エンタープライズ・プライベート・キーは安全なロケーションに保存します。専用の設備の方が安全です。

  サーバ認証の場合、パブリック・エンタープライズ・ルート証明書をクライアントに配布します。

- エンタープライズ・ルート証明書を使用して、ID に署名します。

  パブリック・エンタープライズ・ルート証明書とエンタープライズ・プライベート・キーを使用して、各 ID に署名します。サーバ認証の場合、ID ファイルはサーバ用に使用します。

サードパーティの認証局を使用して、サーバ証明書に署名することもできます。民間認証局は、プライベート・キーを保存するための専用の設備を備えており、高品質なサーバ証明書を作成します。

参照

- 「証明書作成ユーティリティ [createcert]」 975 ページ
- 「グローバル署名証明書」 958 ページ

エンタープライズ・ルート証明書

エンタープライズ・ルート証明書を使用すると、複数のサーバが配備されている環境での、データの整合性と拡張性が向上します。

- 信頼できる証明書を作成するために使用されるプライベート・キーは、専用の設備に保存できます。
- サーバ認証では、クライアントを再設定しなくてもサーバを追加できます。

エンタープライズ・ルート証明書を設定するには、エンタープライズ・ルート証明書と、ID の署名に使用するエンタープライズ・プライベート・キーを作成します。

サーバ証明書の作成の詳細については、「署名付き ID ファイル」 958 ページを参照してください。

エンタープライズ・ルート証明書の生成の詳細については、「グローバル署名証明書」 958 ページを参照してください。

署名付き ID ファイル

エンタープライズ・ルート証明書を使用して、サーバの ID ファイルに署名できます。

サーバ認証の場合、各サーバ用に ID ファイルを生成します。これらの証明書はエンタープライズ・ルート証明書によって署名されるので、createcert -s オプションを使用します。

署名付き ID ファイルの生成の詳細については、「証明書作成ユーティリティ [createcert]」 975 ページを参照してください。

グローバル署名証明書

民間認証局とは、高品質の証明書の作成と、これらの証明書を使用した証明書要求への署名を事業としている組織です。

グローバル署名証明書には、次の利点があります。

- 会社内での通信の場合、共通して信用するものとして、外部の認可された認証局を使用すると、システムのセキュリティの信頼性が高まります。認証局は、署名を行ったすべての証明書の識別情報が正確であることを保証する必要があります。
- 認証局は、証明書を生成するための管理された環境と高度な方法を提供します。

- ルート証明書のプライベート・キーは、秘密にしておきます。企業内ではこの重要情報を格納するのに適した場所がない可能性がありますが、認証局では専用の設備を設計して管理できます。

グローバル署名証明書の設定

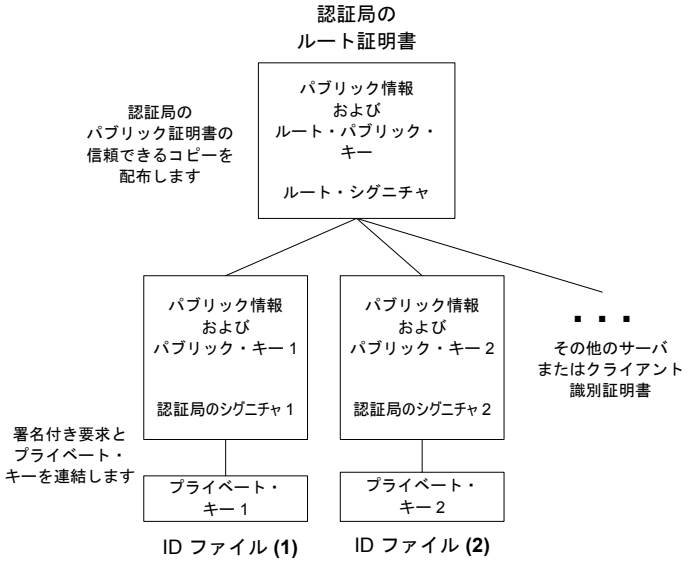
グローバル署名 ID ファイルを設定するには、次の手順に従います。

- -r オプションを指定した createcert ユーティリティを使用して証明書を要求します。「証明書作成ユーティリティ [createcert]」 975 ページを参照してください。
- 認証局を使用して各要求に署名します。署名付き要求と対応するプライベート・キーを組み合わせて、サーバの ID ファイルを作成できます。

エンタープライズ・ルート証明書へのグローバル署名
エンタープライズ・ルート証明書にグローバル署名できます。これは、認証局が、他の証明書に署名できる証明書を生成する場合のみ適用されます。

グローバル署名付き ID ファイルを使用する

グローバル署名証明書を直接サーバの ID ファイルとして使用できます。次の図は、複数の ID ファイルの設定を示します。



dbsrv10 または mlsrv10 コマンド・ラインで、サーバ ID ファイルと、プライベート・キーのパスワードを参照します。

参照

- SQL Anywhere：「トランスポート・レイヤ・セキュリティを使用するデータベース・サーバの起動」 962 ページ
- Mobile Link：「トランスポート・レイヤ・セキュリティを使用する Mobile Link サーバの起動」 968 ページ

認証局の証明書を信用するようにクライアントを設定する

サーバ認証の場合は、サーバにアクセスするクライアントがチェーン内のルート証明書を信頼することを確認する必要があります。グローバル署名証明書の場合、ルート証明書は認証局の証明書です。

証明書フィールドの確認
グローバル署名証明書を使用する場合、各クライアントはフィールド値を確認して、同じ認証局が他のクライアント用に署名した証明書を信用することを避けなければなりません。
「証明書フィールドの確認」 970 ページを参照してください。

サーバ証明書を信用するように Mobile Link クライアントを設定する方法については、「トランスポート・レイヤ・セキュリティを使用する Mobile Link クライアントの設定」 969 ページを参照してください。

トランスポート・レイヤ・セキュリティを使用するデータベース・サーバを構成する方法の詳細については、「トランスポート・レイヤ・セキュリティを使用するデータベース・サーバの起動」 962 ページを参照してください。

グローバル署名証明書を使用して信用を確立する方法については、「グローバル署名証明書」 958 ページを参照してください。

SQL Anywhere クライアント／サーバ通信の暗号化

SQL Anywhere のクライアント／サーバ通信は、トランスポート・レイヤ・セキュリティを使用して暗号化できます。

参照

- 「SQL Anywhere Web サービスでのトランスポート・レイヤ・セキュリティの使用」 966 ページ

トランスポート・レイヤ・セキュリティを使用するデータベース・サーバの起動

トランスポート・レイヤ・セキュリティを使用してデータベース・サーバを起動するには、サーバ ID ファイル名と、サーバのプライベート・キーを保護するパスワードを指定します。

トランスポート・レイヤ・セキュリティを設定する手順の概要については、「トランスポート・レイヤ・セキュリティの設定」 953 ページを参照してください。

-ec サーバ・オプションを使用して、証明書と certificate_password パラメータを指定します。

dbsrv10 コマンド・ラインの一部の構文を次に示します。

```
-ec tls(
  tls_type=cipher;
  certificate=server-identity-filename;
  certificate_password=password )
-x tcpip
```

- ***cipher*** RSA 暗号化の場合は **rsa** を指定し、ECC 暗号化の場合は **ecc** を指定します。FIPS 承認の RSA 暗号化の場合は、**tls_type=rsa;fips=y** を指定します。RSA FIPS は別の承認ライブラリを使用しますが、SQL Anywhere 9.0.2 以降で RSA を使用しているクライアントと互換性があります。

  FIPS がサポートされているプラットフォームのリストについては、「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」に記載されている、SQL Anywhere、Ultra Light、Mobile Link の別途ライセンスが必要なコンポーネントの表を参照してください。

  cipher は、証明書を作成するときに使用される暗号化 (ECC または RSA) と一致する必要があります。

  FIPS 承認のアルゴリズムの実行については、「-fips サーバ・オプション」 166 ページを参照してください。

- ***server-identity-filename*** サーバ ID ファイルのパスとファイル名を指定します。FIPS 承認の RSA 暗号化を使用している場合は、RSA 暗号化を使用して証明書を生成する必要があります。

  サーバ証明書の作成については、「デジタル証明書の作成」 955 ページを参照してください。サーバ証明書は、自己署名証明書、または認証局やエンタープライズ・ルート証明書の署名を受けた証明書のいずれかです。

- ***password*** サーバのプライベート・キーのパスワードを指定します。このパスワードは、サーバ証明書を作成するときに指定します。

単純暗号化を使用してデータベース・サーバを起動することもできますが、データの整合性は保証されず、サーバ認証を行うこともできません。単純暗号化を使用すると、パケット・スニッファを使用して、クライアントとサーバの間で送信されるネットワーク・パケットを読み取るのが困難になります。単純暗号化は、旧バージョンの SQL Anywhere でサポートされています。

-ec サーバ・オプションの詳細については、「-ec サーバ・オプション」 161 ページを参照してください。

TCP/IP プロトコルは、-x サーバ・オプションを使用して指定します。「-x サーバ・オプション」 205 ページを参照してください。

例

次の例では、-ec サーバ・オプションを使用して、ECC セキュリティ、サーバ ID ファイル、サーバのプライベート・キーを保護するパスワードを指定します。

```
dbsrv10 -ec tls(tls_type=ecc;certificate=c:¥test¥serv1_ecc.crt; certificate_password=mypwd) -x tcpip c:¥test¥secure.db
```

設定ファイルとファイル非表示ユーティリティ dbfhide を使用して、パスワードを含むコマンド・ライン・オプションを非表示にできます。「@data サーバ・オプション」 147 ページを参照してください。

次の例では、-ec サーバ・オプションを使用して、RSA セキュリティ、サーバ ID、サーバのプライベート・キーを保護するパスワードを指定します。

```
dbsrv10 -ec tls(tls_type=rsa;certificate=c:¥test¥serv1_rsa.crt; certificate_password=test) -x tcpip c:¥test¥secure.db
```

トランスポート・レイヤ・セキュリティを使用するクライアント・アプリケーションの設定

トランスポート・レイヤ・セキュリティを使用するように、SQL Anywhere クライアント・アプリケーションを設定できます。暗号化接続パラメータ・セットを使用して、信用された証明書、暗号化のタイプ、ネットワーク・プロトコルを指定します。

トランスポート・レイヤ・セキュリティを設定する手順の概要については、「トランスポート・レイヤ・セキュリティの設定」 953 ページを参照してください。

サーバ認証

リモート・クライアントは、サーバ認証を使用することで、データベース・サーバのアイデンティティを確認できます。デジタル署名と証明書フィールドの確認が一緒に機能して、サーバ認証が実現します。

デジタル署名

データベース・サーバ証明書には、データの整合性を維持し、不正侵入を防ぐための、複数のデジタル署名が含まれています。デジタル署名の作成は、次の手順で行われます。

- 証明書で実行されるアルゴリズムが、ユニークな値またはハッシュを生成します。
- 証明書への署名または認証局のプライベート・キーを使用して、ハッシュが暗号化されます。
- デジタル署名と呼ばれる暗号化ハッシュが、証明書に埋め込まれます。

デジタル署名は、自己署名、あるいはエンタープライズ・ルート証明書または認証局の署名を受けています。

クライアント・アプリケーションがデータベース・サーバにアクセスする場合、各クライアントがトランスポート・レイヤ・セキュリティを使用するように設定されていると、サーバはその証明書のコピーをクライアントに送信します。クライアントは、証明書に含まれているサーバのパブリック・キーを使用して証明書のデジタル署名を復号化し、証明書の新しいハッシュを算出して、2 つの値を比較します。値が一致する場合は、サーバの証明書の整合性が確認されます。

FIPS 承認の RSA 暗号化を使用している場合は、RSA を使用して証明書を生成する必要があります。

自己署名証明書の詳細については、「自己署名ルート証明書」 956 ページを参照してください。

エンタープライズ・ルート証明書と認証局の詳細については、「証明書チェーン」 956 ページを参照してください。

証明書フィールドの確認

グローバル署名証明書を使用する場合、各クライアントは証明書のフィールド値を確認して、同じ認証局が他のクライアント用に署名した証明書を信用することを避けなければなりません。この問題を解決するには、証明書の識別情報部分のフィールド値をテストするようにクライアントに要求します。認証局は、署名を行ったすべての証明書の識別情報が正確であることを保証する必要があります。

グローバル署名証明書の詳細については、「グローバル署名証明書」 958 ページを参照してください。

createcert ユーティリティを使用して証明書を作成する場合は、組織、組織単位、通称のフィールドに値を入力します。対応するクライアント接続パラメータを使用して、これらのフィールドを確認します。

- **組織**　組織フィールドは、certificate_company 暗号化接続パラメータに対応します。
- **組織単位**　組織単位フィールドは、certificate_unit 暗号化接続パラメータに対応します。
- **通称**　通称フィールドは、certificate_name 暗号化接続パラメータに対応します。

クライアント側の暗号化接続パラメータの詳細については、「Encryption 接続パラメータ [ENC]」 247 ページを参照してください。

クライアント・セキュリティ・オプション

クライアントは、トランスポート・レイヤ・セキュリティに対して暗号化接続パラメータのセットを使用します。

trusted_certificates パラメータ

これは唯一必須のパラメータです。クライアントは、trusted_certificates 暗号化接続パラメータを使用して、信用されたデータベース・サーバ証明書を指定します。信用された証明書は、サーバの自己署名証明書、パブリック・エンタープライズ・ルート証明書、民間認証局に属する証明書のいずれかです。

詳細については、「デジタル証明書の作成」 955 ページを参照してください。

証明書フィールドの確認

証明書フィールドを確認するには、certificate_company、certificate_unit、certificate_name の各暗号化プロトコル・オプションを使用します。これは、サーバ認証の重要な手順です。サードパーティの認証局を使用して証明書にグローバル署名している場合は、証明書フィールドを確認してください。

証明書フィールドの確認の詳細については、「証明書フィールドの確認」 964 ページを参照してください。

トランスポート・レイヤ・セキュリティを使用するクライアント接続の確立

クライアント・アプリケーションがトランスポート・レイヤ・セキュリティを使用するように設定するには、接続文字列の中で Encryption (ENC) 接続パラメータを使用します。接続文字列の形式は次のとおりです (全体を 1 行で入力してください)。

```
Encryption=tls(
  tls_type=cipher;
  [ fips={ y | n }; ]
  trusted_certificates=public-certificate)
```

- ***cipher*** RSA 暗号化の場合は **rsa** を指定し、ECC 暗号化の場合は **ecc** を指定します。FIPS 承認の RSA 暗号化の場合は、**tls_type=rsa;fips=y** を指定します。RSA FIPS は別の承認ライブラリを使用しますが、SQL Anywhere 9.0.2 以降で RSA を使用しているサーバと互換性があります。

  cipher に指定する暗号化が、証明書を作成するときに使用した暗号化 (RSA または ECC) と一致しない場合、接続は失敗します。

- ***public-certificate*** 信用された証明書を 1 つ以上含むファイルのパスとファイル名を指定します。FIPS 承認の RSA 暗号化を使用している場合は、RSA を使用して証明書を生成する必要があります。

trusted_certificates および他のクライアント・セキュリティ・パラメータの詳細については、「クライアント・セキュリティ・オプション」 965 ページを参照してください。

証明書の作成または取得の詳細については、「デジタル証明書の作成」 955 ページを参照してください。

暗号化接続パラメータの詳細については、「Encryption 接続パラメータ [ENC]」 247 ページを参照してください。

例

次の例では、trusted_certificates 暗号化接続パラメータを使用して、証明書 *public_cert.crt* を指定します。

```
"UID=DBA;PWD=sql;ENG=myeng;LINKS=tcpip;
ENC=tls(tls_type=ecc;trusted_certificates=public_cert.crt)"
```

次の例では、trusted_certificates 暗号化接続パラメータを使用して証明書 *public_cert.crt* を指定し、certificate_unit および certificate_name 暗号化接続パラメータを使用して証明書フィールドを確認します。

```
"UID=DBA;PWD=sql;ENG=myeng;LINKS=tcpip;
ENC=tls(tls_type=ecc;trusted_certificates=public_cert.crt;
certificate_unit=test_unit;certificate_name=my_certificate)"
```

SQL Anywhere Web サービスでのトランスポート・レイヤ・セキュリティの使用

SQL Anywhere Web サービス用にトランスポート・レイヤ・セキュリティを設定するには、次の手順に従います。

- **デジタル証明書を取得する**　サーバ証明書ファイルファイルと ID ファイルが必要です。証明書 (認証局の証明書の場合もあります) は、ブラウザまたは Web クライアントに配布します。サーバ ID ファイルは、SQL Anywhere Web サーバに安全に保存されます。

  認証局の使用に関する情報を含む、デジタル証明書の一般的な情報については、「デジタル証明書の作成」 955 ページを参照してください。

- **トランスポート・レイヤ・セキュリティを指定して Web サーバを起動する**　-xs データベース・サーバ・オプションを使用して、HTTPS、サーバ ID ファイル、プライベート・キーを保護するパスワードを指定します。

  dbsrv10 コマンド・ラインの一部の構文を次に示します。

  -xs *protocol*(
  [**fips=**{ **y** | **n** };]
  Certificate=*server-identity-filename*;
  Certificate_Password=*password*;...) ...

  - ***protocol***　**https**、または FIPS 承認の RSA 暗号化の場合は **https_fips** と **fips=y** を指定します。FIPS 承認の HTTPS は承認された別個のライブラリを使用しますが、HTTPS と互換性があります。

> **注意**
> FIPS 承認の HTTPS を使用する場合は、Mozilla Firefox ブラウザに接続できます。ただし、FIPS 承認の HTTPS が使用する暗号化パッケージ・プログラムは、 Internet Explorer、Opera、または Safari ブラウザではサポートされていません。FIPS 承認の HTTPS を使用する場合、これらのブラウザでは接続できません。

FIPS 承認のアルゴリズムの実行については、「-fips サーバ・オプション」 166 ページを参照してください。

- ***server-identity-filename*** サーバ ID のパスとファイル名を指定します。HTTPS では、RSA 証明書を使用する必要があります。
- ***password*** サーバのプライベート・キーのパスワードを指定します。このパスワードは、サーバ証明書を作成するときに指定します。

-xs サーバ・オプションの詳細については、「-xs サーバ・オプション」 208 ページを参照してください。

certificate パラメータと certificate_password パラメータの詳細については、次の項を参照してください。

- 「Certificate プロトコル・オプション」 267 ページ
- 「Certificate_Password プロトコル・オプション」 268 ページ

- **Web クライアントを設定する** ブラウザまたは他の Web クライアントが証明書を信用するように設定します。信用された証明書は、自己署名証明書、エンタープライズ・ルート証明書、または認証局証明書です。

  認証局の使用に関する情報を含む、デジタル証明書の一般的な情報については、「デジタル証明書の作成」 955 ページを参照してください。

Mobile Link クライアント／サーバ通信の暗号化

Mobile Link のクライアント／サーバ通信は、トランスポート・レイヤ・セキュリティを使用して暗号化できます。

トランスポート・レイヤ・セキュリティを使用する Mobile Link サーバの起動

トランスポート・レイヤ・セキュリティを使用して Mobile Link サーバを起動するには、サーバ証明書と、サーバのプライベート・キーを保護するパスワードを指定します。

トランスポート・レイヤ・セキュリティを設定する手順の概要については、「トランスポート・レイヤ・セキュリティの設定」 953 ページを参照してください。

TCP/IP と HTTPS を使用する Mobile Link サーバの保護

mlsrv10 -x サーバ・オプションを使用して、証明書と certificate_password パラメータを指定します。次に示すのは mlsrv10 コマンド・ラインの一部です (全体を 1 行で入力する必要があります)。

-x *protocol*(
  tls_type=*cipher*;
  fips={ y | n };
  certificate=*server-certificate*;
  certificate_password=*password*;...)

- ***protocol*** **https** または **tls** です。**tls** で指定されるプロトコルは TLS を使用する TCP/IP です。
- ***cipher*** RSA 暗号化の場合は **rsa** を指定し、ECC 暗号化の場合は **ecc** を指定します。cipher は、証明書を作成するときに使用される暗号化と一致する必要があります。
- **fips** RSA 暗号化のみと組み合わせて使用できます。RSA FIPS の場合は、Certicom の別の FIPS 140-2 承認ソフトウェアを使用します。FIPS を使用するサーバは、FIPS を使用しないクライアントと互換性があります。逆についても同様です。RSA FIPS を使用できるのは、サポートされている 32 ビット Windows プラットフォーム上または Solaris 上の SQL Anywhere クライアント、UNIX 上または Windows CE を含むサポートされている 32 ビット Windows プラットフォーム上の Ultra Light クライアントです。
- ***server-certificate*** サーバ証明書のパスとファイル名を指定します。

  サーバ証明書の作成の詳細については、「デジタル証明書の作成」 955 ページを参照してください。サーバ証明書は、自己署名証明書、または認証局やエンタープライズ・ルート証明書の署名を受けた証明書のいずれかです。
- ***password*** サーバ証明書のプライベート・キーのパスワードを指定します。このパスワードは、サーバ証明書を作成するときに指定します。

「-x オプション」 『Mobile Link - サーバ管理』を参照してください。

例

次の例では、セキュリティのタイプ (RSA)、サーバ証明書、サーバのプライベート・キーを保護するパスワードを mlsrv10 コマンド・ラインで指定します。

```
mlsrv10 -c "dsn=my_cons"
 -x tls(tls_type=rsa;certificate=c:¥test¥serv_rsa1.crt;certificate_password=pwd)
```

次の例では、mlsrv10 コマンド・ラインで ECC 証明書を指定します。

```
mlsrv10 -c "dsn=my_cons"
 -x tls(tls_type=ecc;certificate=c:¥test¥serv_ecc1.crt;certificate_password=pwd)
```

mlsrv10 -x オプションの詳細については、「-x オプション」 『Mobile Link - サーバ管理』を参照してください。

サーバ証明書 (この例では *serv1.crt*) の作成については、「デジタル証明書の作成」 955 ページを参照してください。

設定ファイルとファイル非表示ユーティリティ (dbfhide) を使用して、コマンド・ライン・オプションを非表示にできます。「@data オプション」 『Mobile Link - サーバ管理』を参照してください。

トランスポート・レイヤ・セキュリティを使用する Mobile Link クライアントの設定

Mobile Link トランスポート・レイヤ・セキュリティを使用するように SQL Anywhere クライアントまたは Ultra Light クライアントを設定できます。各クライアントに対して、信用された証明書、暗号化のタイプ、ネットワーク・プロトコルを指定します。

トランスポート・レイヤ・セキュリティを設定する手順の概要については、「トランスポート・レイヤ・セキュリティの設定」 953 ページを参照してください。

サーバ認証

リモート・クライアントは、サーバ認証を使用することで、 サーバのアイデンティティを確認できます。デジタル署名と証明書フィールドの確認が一緒に機能して、サーバ認証が実現します。

デジタル署名

サーバ証明書には、データの整合性を維持し、不正侵入を防ぐための、複数のデジタル署名が含まれています。デジタル署名の作成は、次の手順で行われます。

- 証明書で実行されるアルゴリズムが、ユニークな値またはハッシュを生成します。
- 証明書への署名または認証局のプライベート・キーを使用して、ハッシュが暗号化されます。
- デジタル署名と呼ばれる暗号化ハッシュが、証明書に埋め込まれます。

デジタル署名は、自己署名、あるいはエンタープライズ・ルート証明書または認証局の署名を受けています。

Mobile Link クライアントが Mobile Link サーバにアクセスする場合、各クライアントがトランスポート・レイヤ・セキュリティを使用するように設定されていると、サーバはその証明書のコピーをクライアントに送信します。クライアントは、証明書に含まれているサーバのパブリック・キーを使用して証明書のデジタル署名を復号化し、証明書の新しいハッシュを算出して、2つの値を比較します。値が一致する場合は、サーバの証明書の整合性が確認されます。

自己署名証明書の詳細については、「自己署名ルート証明書」 956 ページを参照してください。

エンタープライズ・ルート証明書と認証局の詳細については、「証明書チェーン」 956 ページを参照してください。

証明書フィールドの確認

グローバル署名証明書を使用する場合、各クライアントは証明書のフィールド値を確認して、同じ認証局が他のクライアント用に署名した証明書を信用することを避けなければなりません。この問題を解決するには、証明書の識別情報部分のフィールド値をテストするようにクライアントに要求します。認証局は、署名を行ったすべての証明書の識別情報が正確であることを保証する必要があります。

グローバル署名証明書の詳細については、「グローバル署名証明書」 958 ページを参照してください。

createcert ユーティリティを使用して証明書を作成する場合は、組織、組織単位、通称のフィールドに値を入力します。対応する Mobile Link クライアント接続パラメータを使用して、これらのフィールドを確認します。

- **組織**　組織フィールドは、certificate_company Mobile Link クライアント接続パラメータに対応します。「certificate_company」 『Mobile Link - クライアント管理』を参照してください。
- **組織単位**　組織単位フィールドは、certificate_unit Mobile Link クライアント接続パラメータに対応します。「certificate_unit」 『Mobile Link - クライアント管理』を参照してください。
- **通称**　通称フィールドは、certificate_name Mobile Link クライアント接続パラメータに対応します。「certificate_name」 『Mobile Link - クライアント管理』を参照してください。

Mobile Link クライアントの設定については、次の項を参照してください。

- 「トランスポート・レイヤ・セキュリティを使用する Ultra Light クライアントの設定」 973 ページ
- 「クライアント・セキュリティ・オプション」 970 ページ

デジタル証明書の作成については、「デジタル証明書の作成」 955 ページを参照してください。

クライアント・セキュリティ・オプション

Mobile Link クライアント (SQL Anywhere および Ultra Light) は、共通の接続パラメータ・セットを使用して、トランスポート・レイヤ・セキュリティを設定します。

trusted_certificates パラメータ

Mobile Link クライアントは、trusted_certificates 接続パラメータを使用して、信用された Mobile Link サーバ証明書を指定します。信用された証明書は、サーバの自己署名証明書、パブリック・エンタープライズ・ルート証明書、民間認証局に属する証明書のいずれかです。

次の項を参照してください。

- 「trusted_certificates」『Mobile Link - クライアント管理』
- 「デジタル証明書の作成」 955 ページ

証明書フィールドの確認

証明書フィールドを確認するには、certificate_company、certificate_unit、certificate_name の各接続パラメータを使用します。これは、サーバ認証の重要な手順です。サードパーティの認証局を使用して証明書にグローバル署名している場合は、証明書フィールドを確認してください。

次の項を参照してください。

- 「証明書フィールドの確認」 970 ページ
- 「グローバル署名証明書」 958 ページ
- 「サーバ認証」 969 ページ

トランスポート・レイヤ・セキュリティを使用する SQL Anywhere クライアントの設定

ここでは、HTTPS または TCP/IP でトランスポート・レイヤ・セキュリティを使用する SQL Anywhere クライアントの設定方法について説明します。

TCP/IP および HTTPS におけるトランスポート・レイヤ・セキュリティの使用

Mobile Link トランスポート・レイヤ・セキュリティは、Mobile Link HTTPS および TCP/IP プロトコルの本来の機能です。HTTPS でトランスポート・レイヤ・セキュリティを使用するには、ADR 拡張オプションを使用して、trusted_certificates 接続パラメータを指定します。dbmlsync コマンド・ラインの一部の構文を次に示します。

```
-e "ctp=protocol;
  adr=[ fips={ y | n }; ]
  trusted_certificates=public-certificate;
  ..."
```

- ***protocol*** **https** または **tls** です。**tls** で指定されるプロトコルはトランスポート・レイヤ・セキュリティを使用する TCP/IP です。
- **fips** FIPS 承認の RSA 暗号化用です。FIPS 承認の HTTPS は、Certicom の FIPS 140-2 承認ソフトウェアという別のソフトウェアを使用しますが、HTTPS を使用するバージョン 9.0.2 以降の Mobile Link サーバと互換性があります。
- ***public-certificate*** 信用された証明書のパスとファイル名を指定します。

  HTTPS または FIPS 承認の HTTPS の場合は、RSA 暗号化を使用して作成した証明書を使用してください。

参照

- 「クライアント・セキュリティ・オプション」 970 ページ
- 「デジタル証明書の作成」 955 ページ
- 「CommunicationAddress (adr) 拡張オプション」 『Mobile Link - クライアント管理』
- 「Mobile Link クライアント・ネットワーク・プロトコル・オプション」 『Mobile Link - クライアント管理』
- 「CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」 『SQL Anywhere サーバ - SQL リファレンス』

例

次の例では、HTTPS を使用して RSA セキュリティを指定します。コマンドは、全体を 1 行で入力する必要があります。

```
dbmlsync -c "eng=rem1;uid=dba;pwd=mypwd"
  -e "ctp=https;
      adr=trusted_certificates=c:¥temp¥public_cert.crt;
      certificate_company=Sybase, Inc.;
      certificate_unit=IAS;
      certificate_name=MobiLink)"
```

CREATE SYNCHRONIZATION SUBSCRIPTION 文または ALTER SYNCHRONIZATION SUBSCRIPTION 文を使用して、CommunicationAddress 拡張オプションを指定することもできます。この方法でも同じ情報が提供されますが、その情報はデータベースに保存されます。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
 TO pub1
 FOR user1
 ADDRESS 'trusted_certificates=c:¥temp¥public_cert.crt;
    certificate_company=Sybase, Inc.;
    certificate_unit=IAS;
  certificate_name=MobiLink';
```

次の例では、RSA セキュリティと TCP/IP を指定します。コマンドは、全体を 1 行で入力する必要があります。

```
dbmlsync -c "eng=rem1;uid=myuid;pwd=mypwd"
   -e "ctp=tcpip;
       adr=port=3333;
       tls(
          tls_type=rsa;
          trusted_certificates=c:¥test¥public_cert.crt;
          certificate_company=Sybase, Inc.;
          certificate_unit=IAS;
          certificate_name=MobiLink)"
```

CREATE SYNCHRONIZATION SUBSCRIPTION 文または ALTER SYNCHRONIZATION SUBSCRIPTION 文を使用して、CommunicationAddress 拡張オプションを指定することもできます。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
 TO pub1
 FOR user1
 ADDRESS 'port=3333;
    security=tls(tls_type=rsa;trusted_certificates=public_cert.crt;
```

```
certificate_company=Sybase, Inc.;
certificate_unit=IAS;
certificate_name=MobiLink )';
```

トランスポート・レイヤ・セキュリティを使用する Ultra Light クライアントの設定

Mobile Link トランスポート・レイヤ・セキュリティは、Mobile Link HTTPS プロトコルの本来の機能です。HTTPS および Ultra Light クライアントを使用する場合は、信用された証明書と、ネットワーク・プロトコル・オプションとして、証明書フィールドを直接指定できます。

Ultra Light インタフェースで HTTPS プロトコルを指定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」『Mobile Link - クライアント管理』を参照してください。

tls_type 同期パラメータの詳細については、「tls_type」『Mobile Link - クライアント管理』を参照してください。

♦ TCP/IP または HTTPS でトランスポート・レイヤ・セキュリティを使用するように Ultra Light クライアントを設定するには、次の手順に従います。

1. 信頼されたルート証明書を指定する方法は 2 つあります。

   - **Ultra Light データベースの作成時**　「Ultra Light データベース作成ユーティリティ (ulcreate)」『Ultra Light - データベース管理とリファレンス』または「Ultra Light データベース初期化ユーティリティ (ulinit)」『Ultra Light - データベース管理とリファレンス』を参照してください。
   - **trusted_certificates プロトコル・オプションの使用**　詳細については、手順 3 を参照してください。このオプションは Palm OS では使用できません。

2. TCP/IP または HTTPS プロトコルを同期用に指定します。セキュア TCP/IP のキーワードは tls です。

   次の例は C/C++ Ultra Light で記述されています。tls を指定するには、https を tls に変更します。

   ```
   auto ul_synch_info synch_info;
   conn.InitSynchInfo( &synch_info );
   synch_info.user_name = UL_TEXT( "50" );
   synch_info.version = UL_TEXT( "ul_default" );
   ...
   synch_info.stream = "https";
   ...
   ```

3. TCP/IP または HTTPS プロトコル・オプションを指定します。

   次の例は C/C++ Ultra Light で記述されています。tls を指定するには、https を tls に変更します。

   ```
   auto ul_synch_info synch_info;
   ...
   synch_info.stream = "https";
   synch_info.stream_parms = TEXT(
       "port=9999;
   ```



```
        certificate_company=Sybase, Inc.;
        certificate_unit=IAS;
        certificate_name=MobiLink");
```

certificate_company、certificate_unit、certificate_name の各プロトコル・オプションは、証明書のフィールドを確認するために使用されています。

「証明書フィールドの確認」 970 ページを参照してください。

trusted_certificates HTTPS プロトコル・オプションを指定することもできます。このオプションを指定すると、Ultra Light データベースに埋め込まれている、信用された証明書情報 (手順 1) が上書きされます。trusted_certificates プロトコル・オプションは Palm OS では使用できません。

```
auto ul_synch_info synch_info;
...
synch_info.stream = "https";
synch_info.stream_parms = TEXT(
    "port=9999;
    trusted_certificates=¥rsaroot.crt;
    certificate_company=Sybase, Inc.;
    certificate_unit=IAS;
    certificate_name=MobiLink");
```

HTTPS オプションの詳細については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 『Mobile Link - クライアント管理』を参照してください。

証明書ユーティリティ

ユーザは、一般的にサード・パーティから証明書を購入します。そのような認証局には、証明書を作成するために独自のツールが用意されています。次のツールは、開発用やテスト用の証明書を作成するときに特に便利ですが、稼働時の証明書に使用することもできます。

証明書作成ユーティリティ [createcert]

X.509 証明書を作成します。

構文

createcert [**-r** | **-s**]

オプション	説明
-r	このオプションを使用して、PKCS10 証明書要求を作成します。このオプションを指定すると、署名者など証明書の署名に使用される情報を求めるプロンプトは表示されません。
-s *filename*	このオプションを使用して、指定されたファイル内にある PKCS10 証明書要求に署名します。要求は、DER または PEM でコード化できます。このオプションを指定すると、キー生成やサブジェクト情報を求めるプロンプトは表示されません。

説明

署名済みの証明書を作成するには、オプションなしで createcert を使用します。あるユーザが要求を作成して別のユーザがそれに署名するというように、処理を 2 つのステップに分ける場合、1 番目のユーザは -r を指定した createcert を実行して要求を作成し、2 番目のユーザは -s を指定した createcert を実行して要求に署名することができます。

createcert を実行する場合は、次の情報を求めるプロンプトが表示されます。-r または -s オプションを指定する場合は、一部のプロンプトは表示されません。

- **[暗号化タイプを選択してください]**　このプロンプトは、ECC 暗号化のライセンスを購入した場合にかぎり表示されます。[RSA] または [ECC] を選択します。
- **[RSA のキー長を入力してください (512 ～ 16384)]**　このプロンプトは、RSA 暗号化を選択した場合にかぎり表示されます。512 ～ 16384 ビットの間で長さを選択できます。
- **[ECC 曲線]**　このプロンプトは、ECC 暗号化のライセンスを購入し、かつ ECC 暗号化タイプを選択した場合にかぎり表示されます。ECC 曲線のリストから選択することを求めるプロンプトが表示されます。デフォルトは **sect163k1** です。
- **サブジェクト情報**　エンティティを識別するための次の情報を入力する必要があります。
  - [国コード]
  - [都道府県]

- [地方]
- [組織]
- [組織単位]
- [通称]

- **[署名者の証明書のファイル・パスを入力してください。]** 任意で、署名者の証明書のロケーションとファイル名を指定します。この情報を指定した場合、生成された証明書は署名済み証明書です。この情報を指定しなかった場合、生成された証明書は自己署名ルート証明書です。
- **[署名者のプライベート・キーのファイル・パスを入力してください。]** 証明書要求に関連付けられたプライベート・キーを保存するロケーションとファイル名を指定します。
- **[署名者のプライベート・キーのパスワードを入力してください。]** 任意で、プライベート・キーを暗号化するために使用するパスワードを指定します。パスワードを指定しない場合、プライベート・キーは暗号化されません。
- **[シリアル番号]** 任意で、シリアル番号を指定します。シリアル番号は 40 桁以下の 16 進数文字列である必要があります。この番号は、現在の署名者が署名したすべての証明書でユニークである必要があります。シリアル番号を指定しない場合、シリアル番号として GUID が生成されます。
- **[証明書の有効期間 (年数) (1-100)]** 証明書が有効である年数 (1 ー 100) を指定します。この期間を過ぎると、証明書と、その証明書で署名されたすべての証明書の有効期限が切れます。
- **[認証局 (はい(y) またはいいえ(n))]** この証明書を使用して、他の証明書に署名できるかどうかを示します。デフォルトでは、証明書は認証局ではありません (n)。
- **[キーの使用法]** 証明書のプライベート・キーをどのように使用できるかを示す番号をカンマ区切りのリストで指定します。これは詳細オプションであり、ほとんどの状況ではデフォルト設定で十分です。デフォルトは、証明書が認証局であるかどうかによって異なります。
- **要求を保存するファイル名** このプロンプトは、-r オプションを指定した場合にかぎり表示されます。PCKS10 証明書要求のロケーションとファイル名を指定します。
- **[証明書を保存するファイル名を入力してください。]** 証明書を保存するロケーションとファイル名を指定します。ロケーションとファイル名を指定しなければ、証明書は保存されません。
- **[プライベート・キーを保存するファイル名を入力してください。]** このプロンプトは、-r オプションを指定して、前のプロンプトでファイルを指定した場合にかぎり表示されます。証明書要求に関連付けられたプライベート・キーを保存するロケーションとファイル名を指定します。

  -r オプションを指定しなかった場合は、プライベート・キーを保存するロケーションとファイル名を指定します。ロケーションとファイル名を指定しなければ、プライベート・キーは保存されません。
- **[プライベート・キーを保護するためのパスワードを入力してください。]** 任意で、プライベート・キーを暗号化するために使用するパスワードを指定します。パスワードを指定しない場合、プライベート・キーは暗号化されません。

- **[ID を保存するファイル名を入力してください。]** ID を保存するロケーションとファイル名を指定します。ID ファイルは、証明書、署名者、プライベート・キーの連結です。これがサーバの起動時に指定するファイルです。プライベート・キーが保存されなかった場合は、プライベート・キーを保存するためのパスワードを求めるプロンプトが表示されます。それ以外の場合、先に指定したパスワードが使用されます。ファイル名を指定しなければ、ID は保存されません。ID ファイルを保存しない場合、手動で証明書、署名者、プライベート・キーを ID ファイルに連結できます。

参照

- 「証明書」 952 ページ
- 「証明書ビューワ・ユーティリティ [viewcert]」 978 ページ
- 「-ec サーバ・オプション」 161 ページ
- 「Encryption 接続パラメータ [ENC]」 247 ページ
- 「FIPS 承認の暗号化テクノロジ」 951 ページ

例

次の例では、署名済み証明書を作成します。この例では、署名者の証明書にファイル名を指定しないため、自己署名ルート証明書になります。

```
>createcert
SQL Anywhere X.509 証明書ジェネレータ バージョン 10.0.1.3415
暗号化タイプを選択してください ((R)SA または (E)CC): r
RSA のキー長を入力してください (512 ～ 16384): 1024
キー・ペア作成中...
国コード: CA
都道府県: Ontario
地方: Waterloo
組織: Sybase iAnywhere
組織単位: Engineering
通称: Test Certificate
署名者の証明書のファイル・パスを入力してください:
証明書は自己署名ルートになります。
シリアル番号 [GUID の生成]:
生成されたシリアル番号: bfb89a26fb854955954cabc4d056e177
証明書の有効期間 (年数) (1-100): 10
認証局 (Y/N) [N]:
1. デジタル化シグニチャ
2. 否認防止
3. キーの暗号化
4. データの暗号化
5. キーの承諾
6. 証明書の署名
7. CRL の署名
8. 暗号化のみ
9. 複号化のみ
キーの使用法 [3,4,5]:
証明書を保存するファイル名を入力してください: cert.pem
プライベート・キーを保存するファイル名を入力してください: key.pem
プライベート・キーを保護するためのパスワードを入力してください: pwd
ID を保存するファイル名を入力してください: id.pem
```


証明書ビューワ・ユーティリティ [viewcert]

パブリック・キー・インフラストラクチャ (PKI) オブジェクト内の値の表示、PKI オブジェクトのエンコーディングの変換、プライベート・キーの暗号化と復号化を行います。

構文

viewcert [options] *input-file*

input-file は DER または PEM でコード化された PKI オブジェクトである必要があります。

オプション	説明
-d	このオプションを使用して、出力を DER コード化します。このオプションは、-o オプションと一緒に指定した場合のみ役立ちます。-p とは一緒に使用できません。デフォルトでは、判読可能なテキスト形式で PKI オブジェクトが出力されます。
-ip *input-password*	*input-file* に暗号化されたプライベート・キーが含まれる場合、このオプションを使用して、復号化に必要なパスワードを指定します。
-o *output-file*	このオプションを使用して、出力が書き込まれるファイルを指定します。デフォルトでは、出力はコンソールに書き込まれます。
-op *output-password*	このオプションを使用して、プライベート・キーを暗号化するために使用されるパスワードを指定します。このオプションは、-d または -p と一緒に指定した場合のみ役立ちます。デフォルトでは、プライベート・キーは暗号化されません。
-p	このオプションを使用して、出力を PEM コード化します。このオプションは、-o オプションと一緒に指定した場合のみ役立ちます。-d とは一緒に使用できません。デフォルトでは、判読可能なテキスト形式で PKI オブジェクトが出力されます。

説明

viewcert ユーティリティを使用して表示できる PKI オブジェクトのタイプは、次のとおりです。

- X.509 証明書
- 証明書要求
- プライベート・キー
- 証明書失効リスト (CRL)

viewcert は、DER と PEM の間でコード化タイプを変換したり、プライベート・キーを暗号化または復号化したりするために使用することもできます。

viewcert ユーティリティは、RSA オブジェクトと ECC オブジェクトをサポートします。ECC オブジェクトを表示するには、別のライセンスを注文する必要があります。「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

参照

- 「証明書作成ユーティリティ [createcert]」 975 ページ

例

次の例では、SQL Anywhere に付属するサンプル RSA 証明書を表示できます。

```
viewcert rsaroot.crt
```

この例は、次の出力を生成します。

```
SQL Anywhere X.509 Certificate Viewer Version 10.0.1.3330

SQL Anywhere X.509 証明書ビューワ バージョン 10.0.1.3415

X.509 証明書
------------
通称:                RSA Root
組織単位:              test
組織:                test
地方:                test
都道府県:              test
国コード:              test
発行元:               RSA Root
シリアル番号:            303031
発行済み:              Apr 16, 2002   1:53:51
失効:                Apr 17, 2022   1:53:51
シグニチャ・アルゴリズム:RSA, MD5
キー・タイプ:            RSA
キー・サイズ:            1024 bits
基本的な制約:            認証局であり、パス長制限は次のとおり : 10
キーの使用法:            証明書の署名, CRL の署名
```

証明書生成ユーティリティ [gencert] (旧式)

新しい ECC 証明書または RSA 証明書を作成したり、事前に生成した証明書要求に署名したりします。このユーティリティは使用されなくなりました。「証明書作成ユーティリティ [createcert]」 975 ページを使用してください。

構文

gencert [-c | -s] [-r | -q]

オプション	説明
-c	他の証明書の署名に使用できる証明書を作成します。-r とともに使用すると、エンタープライズ・ルート証明書が生成されます。
-s	サーバ ID ファイルを作成します。サーバ ID ファイルは、サーバのプライベート・キーと証明書から成ります。Mobile Link サーバを起動するとき (Mobile Link トランスポート・レイヤ・セキュリティの場合)、またはデータベース・サーバを起動するとき (SQL Anywhere クライアント／サーバのトランスポート・レイヤ・セキュリティの場合)、サーバ ID ファイルが参照されます。-r とともに使用すると、自己署名証明書が生成されます。

オプション	説明
-r	自己署名ルート証明書を作成します。-s とともに使用すると、自己署名証明書が作成されます。-c とともに使用すると、他の証明書の署名に使用できるエンタープライズ・ルート証明書が作成されます。他のオプションは使用せずに gencert -r を指定すると、サーバ証明書またはエンタープライズ・ルートとして使用できる証明書が作成されます。このオプションは、-q との互換性はありません。
-q *request-file*	事前に生成した証明書要求に署名します。-s とともに使用すると、サーバ証明書が作成されます。-c とともに使用すると、他の証明書の署名に使用できるエンタープライズ・ルート証明書が作成されます。他のオプションは使用せずに gencert -q を指定すると、サーバ証明書またはエンタープライズ・ルートとして使用できる証明書が作成されます。-q オプションは、-r オプションとの互換性はありません。

-s または -c を指定しない場合、証明書には両方のオプションで提供される機能が含まれます。つまり、この証明書は、他の証明書の署名で使用することも、サーバ証明書として直接使用することもできます。

説明

gencert ユーティリティを使用すると、Mobile Link 同期または SQL Anywhere クライアント／サーバ通信を保護するために使用される、信頼できる証明書、プライベート・キー、サーバ証明書を生成できます。このユーティリティは、各種のセキュリティ設定に対して、X509 証明書 (標準の証明書形式) を作成します。

gencert ユーティリティでは、次の情報の入力が要求されます。

- **[暗号化]**　ECC 暗号化または RSA 暗号化の選択が要求されます。ECC 証明書を生成する場合は、gencert は ECC キー・ペアを生成します。RSA 証明書を生成する場合、certutil10 は 512 ー 2048 の範囲でキー・サイズを指定するように要求し、RSA を使用して証明書を作成します (一般に、キーが長くなるほど暗号化は強力になりますが、処理時間が長くなります)。

  どちらの暗号化を選択する場合も、サーバとクライアントを起動するときに、その暗号化を指定してください。ECC 証明書の場合は tls_type=ecc を指定します。RSA 証明書の場合は tls_type=rsa または tls_type=rsa;fips=yes を指定します。

- **[国]、[都道府県]、[地方]**　これらの値は、一般的な証明書識別情報を提供します。グローバル署名証明書を使用する予定の場合は、サード・パーティの認証局によって地方フィールドも要求されます。

  認証局の使用の詳細については、「グローバル署名証明書」 958 ページを参照してください。

- **[組織]、[組織単位]、[通称]**　これらのフィールドによって、クライアントが正しい証明書を認証していることがさらに保証されます。クライアント側では、これらは certificate_company、certificate_unit、certificate_name というプロトコル・オプションにそれぞれ対応します。

「証明書フィールドの確認」 970 ページを参照してください。

- **[シリアル番号]** 証明書のシリアル番号の選択が要求されます。シリアル番号は英数字です。
- **[証明書の有効期間 (年数)]** 証明書の有効期間 (年数) が要求されます。証明書が失効すると、その証明書によって署名されたすべての証明書も失効します。指定した期間が過ぎると、エンタープライズ・ルート、各サーバ証明書、クライアントに配布される証明書を再生成する必要があります。
- **[プライベート・キーを保護するためのパスワードを入力してください。]** certificate_password プロトコル・オプションで指定するパスワードです。
- **[証明書を保存するファイル名を入力してください。]** 証明書のファイル名とロケーションを選択します。
- **[プライベート・キーを保存するファイル名を入力してください。]** プライベート・キーのファイル名とロケーションを選択します。
- **[サーバ ID を保存するファイル名を入力してください。]** サーバ証明書のファイル名とロケーションを選択します。

参照

- 「証明書」 952 ページ
- 「証明書読み込みユーティリティ [readcert] (旧式)」 981 ページ
- 「-ec サーバ・オプション」 161 ページ
- 「Encryption 接続パラメータ [ENC]」 247 ページ
- 「FIPS 承認の暗号化テクノロジ」 951 ページ

証明書読み込みユーティリティ [readcert] (旧式)

readcert ユーティリティは、証明書内の値を表示し、証明書のチェーンを検証します。このユーティリティは使用されなくなりました。「証明書ビューワ・ユーティリティ [viewcert]」 978 ページを使用してください。

構文

readcert *certificate-name*

説明

指定できる証明書は ECC または RSA です。

ECC または RSA 同期ストリームを介して同期が発生すると、Mobile Link サーバは、証明書をクライアントに送信します。送信する証明書はエンティティの署名付き証明書だけでなく、自己署名ルート証明書にまで至ります。クライアントは、チェーンが有効であることとチェーン内のルート証明書が信頼できることを確認します。

このユーティリティは X.509 認証証明書をスキャンし、フィールド値を表示します。その後、証明書のチェーンが有効であることを確認します。チェーン内のいずれかの証明書の期限が切れていたり、順序が正しくなかったり、見つからなかったりした場合、検証エラーが表示されます。

参照

- 「証明書生成ユーティリティ [gencert] (旧式)」 979 ページ

パート VII. レプリケーション

パート VII では、SQL Anywhere を Open Server として使用する方法と、Replication Server によってデータをレプリケートする方法について説明します。

第 25 章

Open Server としての SQL Anywhere

目次

Open Client、Open Server、TDS

SQL Anywhere は、クライアント・アプリケーションにとっては Open Server として機能します。この機能を使用すると、Sybase Open Client アプリケーションは SQL Anywhere データベースにネイティブに接続できます。

単に、Sybase のアプリケーションを SQL Anywhere とともに使用するだけの場合は、Open Client、Open Server、または TDS の詳細を知る必要はありません。しかし、これらのコンポーネントがどのように互いに適合するかを理解すれば、データベースの設定やアプリケーションの設定に役立ちます。この項では、これらのコンポーネントが互いにどう適合するかを説明しますが、それぞれの内部機能についての説明は省略します。

Open Client と Open Server

SQL Anywhere とその他の Adaptive Server ファミリのメンバは、「**Open Server**」として動作します。つまり、Sybase から入手可能な「**Open Client**」ライブラリを使用して、クライアント・アプリケーションを開発できます。Open Client には、Client Library (CT-Library) インタフェース、旧式の DB-Library インタフェースの両方が含まれています。

Open Client アプリケーションを開発して SQL Anywhere とともに使用する方法については、「Sybase Open Client API」『SQL Anywhere サーバ - プログラミング』を参照してください。

Tabular Data Stream

Open Client と Open Server は、「**Tabular Data Stream**」(TDS) と呼ばれるアプリケーション・プロトコルを使用して情報を交換します。Sybase Open Client ライブラリを使用して構築されたすべてのアプリケーションは、TDS アプリケーションでもあります。これは、Open Client ライブラリが TDS インタフェースを使用するためです。ただし、(jConnect などの) 一部のアプリケーションは、Sybase Open Client ライブラリを使用しませんが、TDS アプリケーションです。これらのアプリケーションは、TDS プロトコルを使用して直接通信します。

多くの Open Server が Sybase Open Server ライブラリを使用して TDS へのインタフェースを処理していますが、固有の TDS への直接インタフェースを持つアプリケーションもあります。Sybase の Adaptive Server Enterprise と SQL Anywhere には、両方とも内部 TDS インタフェースがあります。両方ともクライアント・アプリケーションには Open Server として表示されますが、Sybase Open Server ライブラリを使用しません。

プログラミング・インタフェースとアプリケーション・プロトコル

SQL Anywhere は 2 種類のアプリケーション・プロトコルをサポートします。Open Client アプリケーションと Sybase アプリケーション (Replication Server、OmniConnect など) では、TDS を使用します。ODBC アプリケーションと Embedded SQL アプリケーションでは、それぞれ個別に、SQL Anywhere に特有のアプリケーション・プロトコルを使用します。

TDS による TCP/IP の使用

TDS のようなアプリケーション・プロトコルは、ネットワーク・トラフィックを処理する下位レベルの通信プロトコルの一番上に位置します。SQL Anywhere は、TCP/IP ネットワーク・プロトコル上でのみ TDS をサポートします。それに対し、SQL Anywhere 固有のアプリケーション・

プロトコルは、同一コンピュータでの通信用に設計された共有メモリ・プロトコルだけでなく、さまざまなネットワーク・プロトコルをサポートします。

Sybase アプリケーションと SQL Anywhere

SQL Anywhere を Open Server として動作できるので、Replication Server や OmniConnect などの Sybase アプリケーションを SQL Anywhere とともに動作させることができます。

Replication Server のサポート

Open Server インタフェースによって、Sybase Replication Server をサポートできます。つまり、Replication Server が Open Server インタフェースを介して接続することによって、SQL Anywhere データベースを Replication Server インストール環境のレプリケート・サイトとして動作させることができます。

データベースを Replication Server インストール環境のプライマリ・サイトとして動作させるには、「**Log Transfer Manager**」とも呼ばれる Sybase SQL Anywhere 用の Replication Agent も使用してください。

Replication Agent については、「Replication Server を使用したデータのレプリケート」 999 ページを参照してください。

OmniConnect のサポート

Sybase OmniConnect は、データの内容やデータの所在がわからなくても複数のデータ・ソースにアクセスすることにより、企業内でデータを統合して表示することができます。さらに、OmniConnect は、企業全体に渡ってデータの異機種間ジョインを実行し、DB2、Sybase Adaptive Server Enterprise、Oracle、VSAM のようなターゲットについて、プラットホームを問わないテーブル・ジョインを可能にします。

Open Server インタフェースを使用すると、SQL Anywhere を OmniConnect 用のデータ・ソースとして使用できます。

SQL Anywhere を Open Server として設定する

この項では、Open Client アプリケーションからの接続を受信するように SQL Anywhere サーバを設定する方法を説明します。

システムの稼働条件

SQL Anywhere を Open Server として使用するには、クライアント側とサーバ側でそれぞれ別の稼働条件があります。

サーバ側の稼働条件

SQL Anywhere を Open Server として使用するには、サーバ側に次の要素が必要です。

- **SQL Anywhere サーバ・コンポーネント**　ネットワークを介して Open Server にアクセスする場合は、ネットワーク・サーバ (*dbsrv10.exe*) を使用します。パーソナル・サーバ (*dbeng10.exe*) を Open Server として使用できるのは、同一コンピュータからの接続に限られます。
- **TCP/IP**　ネットワーク接続をしていない場合でも、SQL Anywhere を Open Server として使用するには、TCP/IP プロトコル・スタックが必要です。

クライアント側の稼働条件

Sybase クライアント・アプリケーションを使用して SQL Anywhere を含む Open Server に接続するには、次の要素が必要です。

- **Open Client コンポーネント**　アプリケーションが Open Client を使用する場合、TDS 経由でアプリケーションが通信するために必要なネットワーク・ライブラリは、Open Client ライブラリによって提供されます。
- **jConnect**　アプリケーションが JDBC を使用する場合は、jConnect と Java ランタイム環境が必要です。SQL Anywhere は、jConnect 5.5 と 6.0.5 をサポートしています。これらのバージョンは http://www.sybase.com/products/informationmanagement/softwaredeveloperkit/jconnect から入手できます。
- **DSEdit**　サーバ名を Open Client アプリケーションから使用できるようにするには、ディレクトリ・サービス・エディタ DSEdit が必要です。UNIX プラットフォームでは、このユーティリティは sybinit と呼ばれます。

  DSEdit は、SQL Anywhere には付属していませんが、Open Server ソフトウェアには付属しています。

データベース・サーバを Open Server として起動する

SQL Anywhere を Open Server として使用する場合は、TCP/IP プロトコルを使用して起動してください。デフォルトでは、使用可能なすべての通信プロトコルがサーバによって起動されます

が、起動されるプロトコルをコマンドに明示的にリストすることによって、それらのプロトコルを制限できます。たとえば、次のコマンドは両方とも有効です。

```
dbsrv10 -x tcpip,spx c:¥mydata.db
dbsrv10 -x tcpip -n myserver c:¥mydata.db
```

最初のコマンドでは、TCP/IP プロトコルと SPX プロトコルの両方を使用します。ここでは、TCP/IP は Open Client アプリケーションによって使用されます。2 番目のコマンド・ラインでは、TCP/IP だけを使用します。

パーソナル・データベース・サーバは TCP/IP プロトコルをサポートするため、このサーバを、同一コンピュータ上の通信用 Open Server として使用できます。

このサーバは、TDS 経由で Open Client アプリケーションにサービスすると同時に、SQL Anywhere 専用のアプリケーション・プロトコルを使用して、TCP/IP プロトコルなどのプロトコル経由で他のアプリケーションにサービスを提供します。

ポート番号

あるコンピュータ上で TCP/IP を使用するすべてのアプリケーションは、それぞれ別の TCP/IP「**ポート**」を使用するので、ネットワーク・パケットは正しいアプリケーションに到達します。SQL Anywhere のデフォルトのポート番号は 2638 です。SQL Anywhere は Internet Adapter Number Authority (IANA) によってこのポート番号を付与されているので、デフォルトのポート番号を使用することをおすすめします。別のポート番号を使用する場合は、ServerPort (PORT) プロトコル・オプションを使用して番号を指定します。

```
dbsrv10 -x tcpip(ServerPort=2629) -n myserver c:¥mydata.db
```

複数のローカル・データベース・サーバが稼働している場合、またはネットワーク・サーバに接続する場合には、EngineName も指定する必要があります。

Open Client の設定値

このサーバに接続するには、データベース・サーバを実行しているコンピュータ名とそのサーバが使用している TCP/IP ポートをクライアント・コンピュータ側の interfaces ファイルに指定します。

クライアント・コンピュータの設定の詳細については、「Open Server の設定」 990 ページを参照してください。

Open Server の設定

SQL Anywhere は、ネットワーク上の他の Adaptive Server アプリケーション、Open Server アプリケーション、クライアント・ソフトウェアと通信できます。クライアントは 1 つ以上のサーバと通信でき、サーバはリモート・プロシージャ・コールを通して他のサーバと通信できます。製品が互いに対話するには、他の製品がネットワークのどこに常駐しているかをそれぞれが認識する必要があります。このネットワーク・サービス情報は interfaces ファイルに格納されています。

interfaces ファイル

通常、**interfaces ファイル**の名前は、PC オペレーティング・システムでは *SQL.ini*、UNIX オペレーティング・システムでは *interfaces* または *interfac* です。

interfaces ファイルは、住所録のようなもので、コンピュータ上の Open Client アプリケーションが認識しているあらゆるデータベース・サーバの名前とアドレスが列記されています。Open Client プログラムを使用してデータベース・サーバに接続すると、プログラムは interfaces ファイルでサーバの名前を検索し、そのアドレスを使用してサーバに接続します。

interfaces ファイルの名前、場所、内容はオペレーティング・システムによって異なります。また、interfaces ファイル中のアドレスのフォーマットもネットワーク・プロトコルによって異なります。

SQL Anywhere をインストールすると、インストーラによって簡単な interfaces ファイルが作成されます。このファイルを使用して、TCP/IP 経由で SQL Anywhere にローカル接続できます。システム管理者の役割は、interfaces ファイルを修正してユーザに配布し、ユーザがネットワークを通じて SQL Anywhere に接続できるようにすることです。

DSEdit ユーティリティの使用

DSEdit ユーティリティは Windows ユーティリティです。このユーティリティを使用して interfaces ファイル (*SQL.ini*) を設定できます。以下の項では、DSEdit ユーティリティを使用して interfaces ファイルを設定する方法を説明します。

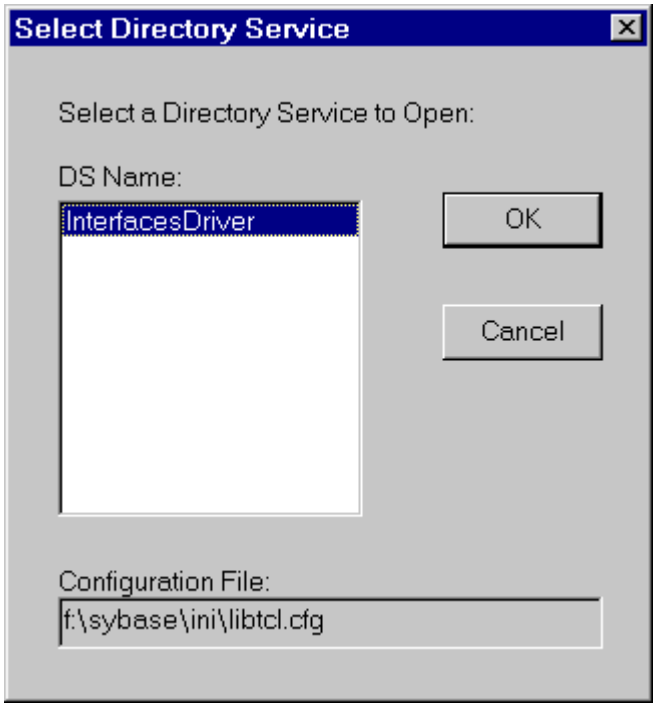
以下の項では、SQL Anywhere に必要なタスクのために DSEdit を使用する方法を説明します。これは、DSEdit ユーティリティの完全なマニュアルではありません。

DSEdit の詳細については、他の Sybase 製品に同梱されているプラットフォーム別の『*ユーティリティ・プログラム*』マニュアルを参照してください。

DSEdit の起動

DSEdit 実行プログラムは、*SYBASE¥bin* ディレクトリにあります。このディレクトリは、インストール時にパスに追加されます。DSEdit は、コマンド・プロンプトまたはエクスプローラから通常の方法で起動できます。

DSEdit を起動すると、[Select Directory Service] ダイアログが表示されます。



ディレクトリ・サービスのセッションを開く

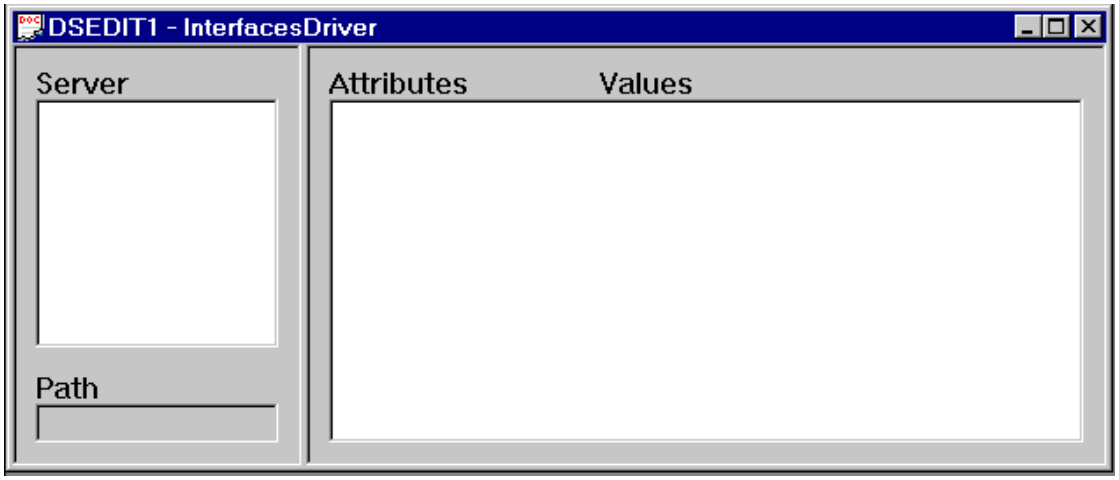
[Select Directory Service] ウィンドウで、ディレクトリ・サービスのセッションを開きます。セッションを開くと、interfaces ファイル (*SQL.ini*)、または *libtcl.cfg* ファイルにリストされたドライバを持つディレクトリ・サービスを編集できます。

♦ **セッションを開くには、次の手順に従います。**

1. [DS name] ボックスのリストの中から、接続先とするディレクトリ・サービスのローカル名をクリックし、[OK] をクリックします。

   SQL Anywhere の場合は、[Interfaces Driver] を選択してください。

SYBASE 環境変数の設定が必要です
DSEdit ユーティリティは、SYBASE 環境変数を使用して *libtcl.cfg* ファイルを検索します。SYBASE 環境変数が正しくない場合、DSEdit は *libtcl.cfg* ファイルを見つけることができません。



このウィンドウで、SQL Anywhere サーバなどのサーバについて、エントリの追加、修正、削除ができます。

サーバ・エントリの追加

♦ **サーバ・エントリを追加するには、次の手順に従います。**

1. [Server Object] – [Add] を選択します。

   [サーバ名入力] ウィンドウが表示されます。

2. [Server Name] ボックスにサーバ名を入力し、[OK] をクリックしてサーバ名を入力します。

   入力するサーバ名は、接続先のデータベース名と一致する必要があります。サーバ・*アドレス*は、サーバの名前と場所を指定するために使用されます。サーバ名のフィールドは、Open Client の識別子です。SQL Anywhere では、サーバに複数のデータベースがロードされている場合、DSEdit サーバ名エントリによって使用するデータベースが識別されます。

   サーバ・エントリが [Server] ボックスに表示されます。サーバの属性を指定するには、エントリを修正します。

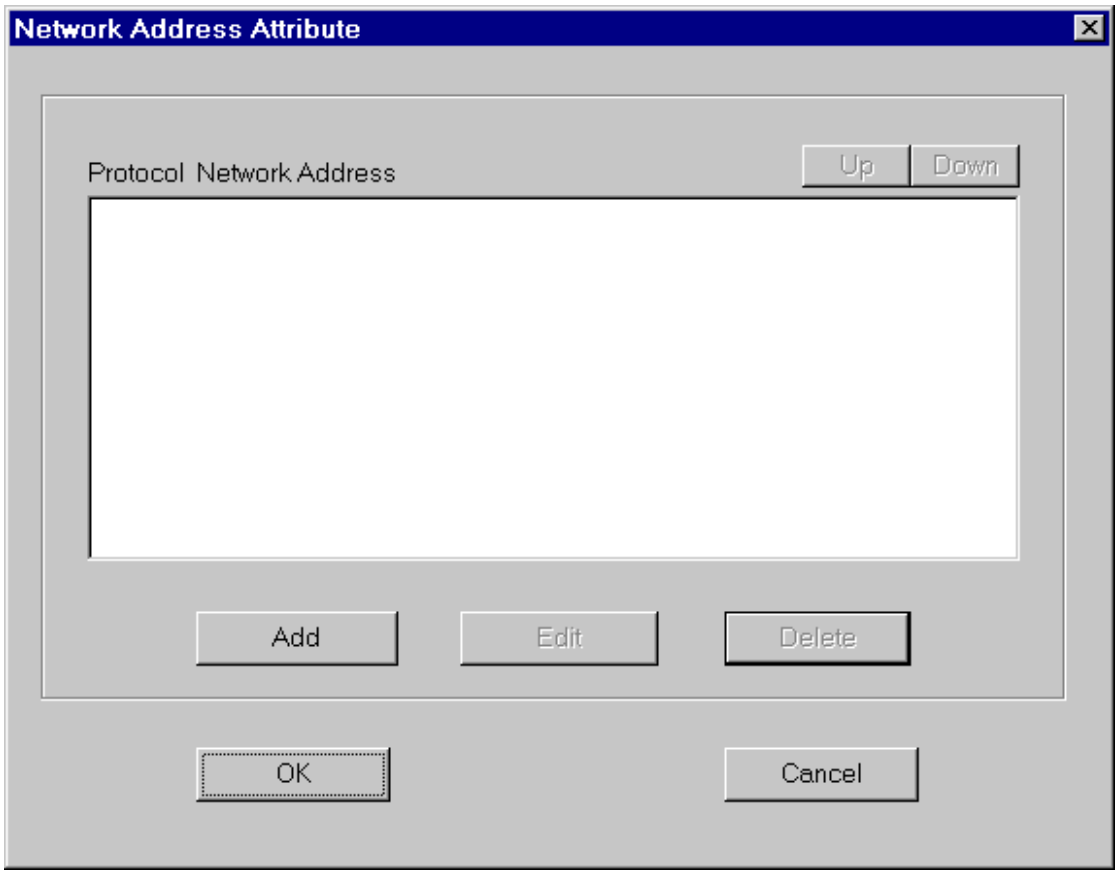
サーバ・アドレスの追加または変更

サーバ名を入力した場合、サーバ・アドレスを修正して interfaces ファイル入力を完了する必要があります。

♦ **サーバ・アドレスを入力するには、次の手順に従います。**

1. [サーバ] ボックスで、サーバ・エントリを選択します。
2. [Attributes] ボックスで、サーバ・アドレスを右クリックします。

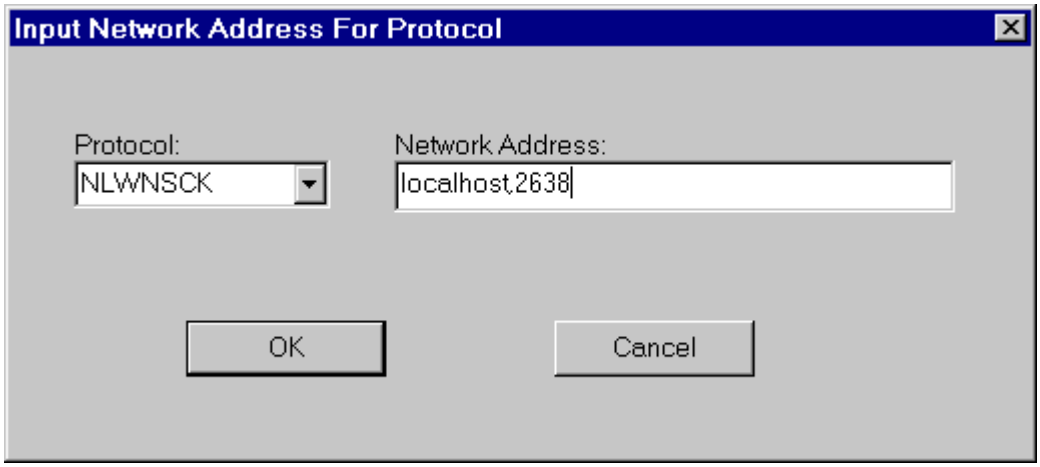
3. ポップアップ・メニューから [Modify Attribute] を選択します。ウィンドウが表示され、アドレスの現在の値を表示します。アドレスを 1 つも入力しなかった場合は、ボックスは空になります。



4. [Add] をクリックします。

   [Input Network Address for Protocol] ウィンドウが表示されます。

5. [プロトコル] ドロップダウン・リストから、TCP/IP プロトコルの [NLWNSCK] を選択し、[ネットワーク・アドレス] フィールドに値を入力します。



TCP/IP アドレスは、次のいずれかの形式で指定します。

- コンピュータ名、ポート番号
- IP アドレス、ポート番号

アドレス (またはコンピュータ名) とポート番号をカンマで区切ります。

マシン名

サーバを実行しているコンピュータは、名前 (または IP アドレス) によって識別されます。Windows オペレーティング・システムの場合、[コントロール パネル] の [ネットワーク] でコンピュータ名を検索できます。

クライアントとサーバが同一のコンピュータ上にある場合でも、コンピュータ名を入力します。この場合は、**localhost** を使用して現在のコンピュータを識別できます。

ポート番号

入力するポート番号は、SQL Anywhere データベース・サーバを実行させるために使用したポート番号と一致させます。「データベース・サーバを Open Server として起動する」 988 ページを参照してください。

SQL Anywhere サーバのデフォルトのポート番号は 2638 です。この番号は、Internet Adapter Number Authority によって SQL Anywhere に割り当てられており、明示的に他のポートを使用する正当な理由がないかぎり、このポートを使用することをおすすめします。

次に示すのは、有効なサーバ・アドレス・エントリです。

```
elora,2638
123.85.234.029,2638
```

サーバ・アドレスの確認

[サーバ・オブジェクト] メニューから ping コマンドを使用して、ネットワーク接続を確認できます。

データベース接続は検証されません
ネットワーク接続を検証すると、指定されたコンピュータ名とポート番号でサーバが要求を受信していることが確認されます。データベース接続については何も確認されません。

♦ **サーバを ping するには、次の手順に従います。**

1. データベース・サーバが実行していることを確認します。
2. DSEdit セッション・ウィンドウの [サーバ] ボックスで、サーバ・エントリをクリックします。
3. [サーバ・オブジェクト] メニューから [Ping Server] を選択します。

   [ping] ウィンドウが表示されます。
4. ping するアドレスを選択します。[ping] をクリックします。

   メッセージ・ボックスが表示され、接続できたかどうかが通知されます。接続できたことが表示された場合、オープン接続とクローズ接続の両方に成功したことを意味します。

サーバ・エントリ名の変更

DSEdit セッション・ウィンドウからサーバ・エントリ名を変更できます。

♦ **サーバ・エントリ名を変更するには、次の手順に従います。**

1. [サーバ] ボックスで、サーバ・エントリを選択します。
2. [サーバ・オブジェクト] – [名前の変更] を選択します。

   [サーバ名入力] ウィンドウが表示されます。
3. [サーバ名] ボックスで、新しいサーバ・エントリ名を入力します。
4. [OK] をクリックして、変更を確定します。

サーバ・エントリの削除

DSEdit セッション・ウィンドウからサーバ・エントリを削除できます。

♦ **サーバ・エントリを削除するには、次の手順に従います。**

1. [サーバ] ボックスで、サーバ・エントリをクリックします。

2. [サーバ・オブジェクト] – [削除] を選択します。

JDBC のサーバの設定

JDBC 接続のアドレス (URL) には、サーバ検索用の必要な情報がすべて含まれています。「ドライバへの URL の指定」『SQL Anywhere サーバ - プログラミング』を参照してください。

Open Client と jConnect 接続の特性

SQL Anywhere は、TDS を通してアプリケーションをサービスしている場合、関連したデータベースのさまざまなオプションを自動的に設定し、オプションの値を Adaptive Server Enterprise のデフォルトの設定と互換性があるようにします。このオプションの設定は、その接続中だけの一時的なものです。オプションは、クライアント・アプリケーションによっていつでも無効にできます。

デフォルトの設定値

TDS を使用する接続で設定されるデータベース・オプションは、次のとおりです。

オプション	設定値
allow_nulls_by_default	Off
ansi_blanks	On
ansinull	Off
automatic_timestamp	On
chained	Off
close_on_endtrans	Off
date_format	YYYY-MM-DD
date_order	MDY
escape_character	Off
float_as_double	On
isolation_level	1
on_tsql_error	Continue
quoted_identifier	Off
time_format	HH:NN:SS.SSS
timestamp_format	YYYY-MM-DD HH:NN:SS.SSS
tsql_hex_constant	On
tsql_variables	On

起動オプションの設定方法

TDS 接続用のデフォルト・データベース・オプションは、システム・プロシージャ sp_tsql_environment を使用して設定されます。このプロシージャでは、以下のオプションを設定します。


```
SET TEMPORARY OPTION allow_nulls_by_default='Off';
SET TEMPORARY OPTION ansi_blanks='On';
SET TEMPORARY OPTION ansinull='Off';
SET TEMPORARY OPTION automatic_timestamp='On';
SET TEMPORARY OPTION chained='Off';
SET TEMPORARY OPTION close_on_endtrans='Off';
SET TEMPORARY OPTION date_format='YYYY-MM-DD';
SET TEMPORARY OPTION date_order='MDY';
SET TEMPORARY OPTION escape_character='Off';
SET TEMPORARY OPTION float_as_double='On';
SET TEMPORARY OPTION isolation_level='1';
SET TEMPORARY OPTION on_tsql_error='Continue';
SET TEMPORARY OPTION quoted_identifier='Off';
SET TEMPORARY OPTION time_format='HH:NN:SS.SSS';
SET TEMPORARY OPTION timestamp_format='YYYY-MM-DD HH:NN:SS.SSS';
SET TEMPORARY OPTION tsql_hex_constant='On';
SET TEMPORARY OPTION tsql_variables='On';
```

sp_tsql_environment プロシージャは編集しないでください
sp_tsql_environment プロシージャは、自分で変更しないでください。このプロシージャは、システムが専用に使用します。

このプロシージャは、TDS 通信プロトコルを使用する接続についてのみ、オプションを設定します。設定される接続には、jConnect を使用する Open Client 接続や JDBC 接続があります。その他の接続 (ODBC と Embedded SQL) は、データベース用のデフォルト設定値を持っています。

TDS 接続用のオプションは、次のようにして変更できます。

♦ **TDS 接続用オプションの設定値を変更するには、次の手順に従います。**

1. 目的のデータベース・オプションを設定するプロシージャを作成します。たとえば、次のようなプロシージャを使用できます。

   ```
   CREATE PROCEDURE my_startup_procedure()
   BEGIN
     IF CONNECTION_PROPERTY('CommProtocol')='TDS' THEN
       SET TEMPORARY OPTION quoted_identifier='Off';
     END IF
   END;
   ```

   このプロシージャの例では、デフォルト設定の quoted_identifier オプションだけを変更します。

2. login_procedure オプションに新しいプロシージャ名を設定します。

   ```
   SET OPTION login_procedure= 'DBA.my_startup_procedure';
   ```

今後の接続では、このプロシージャを使用します。別のユーザ ID に対して、別のプロシージャを設定できます。

データベース・オプションの詳細については、「データベース・オプション」 405 ページを参照してください。

第 26 章

Replication Server を使用したデータのレプリケート

目次

SQL Anywhere と Replication Server の併用に関する概要

Replication Server は、トランザクションの双方向レプリケーションを行う接続ベースのテクノロジです。高速ネットワークで接続された少数の企業データベース間でレプリケーションを行う場合に適しています。通常は、各サイトに管理者がいます。このような設定では、タイムラグを数秒程度に抑えることが可能です。

Mobile Link と SQL Remote を使用して SQL Anywhere データをレプリケートすることもできます。

次の項を参照してください。

- 「Mobile Link 同期について」 『Mobile Link - クイック・スタート』
- SQL Remote 『SQL Remote』

始める前に

この章は、SQL Anywhere を Replication Server インストール環境にセット・アップする Replication Server 管理者を対象にしています。管理者は、Replication Server のマニュアルを読み、Replication Server 製品についての知識を持っておく必要があります。この章では、Replication Server 自体についての説明はありません。

Replication Server の設計、コマンド、管理などの詳細については、Replication Server のマニュアルを参照してください。

注意
SQL Anywhere には、Replication Server システムで SQL Anywhere データベースを使用できるようにするコンポーネントが付属しています。Replication Server は SQL Anywhere インストール環境には含まれていません。

別途ライセンスが必要な必須オプション
Log Transfer Manager (LTM) は、Sybase Replication Server 用の SQL Anywhere Replication Agent であり、SQL Anywhere データベースが Sybase Replication Server インストール環境にプライマリ・サイトとして参加する場合に必要となります。LTM 用のライセンスは、別途注文する必要があります。SQL Anywhere をレプリケート・サイトとして使用する場合は、LTM を使用する必要はありません。
詳細については、「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

Replication Server の特徴

Replication Server は、レプリケーション・システム用に設計されたもので、以下の要件があります。

- **少数のデータベース** Replication Server は、サーバ間のレプリケーションをサポートするように設計されており、通常、1 つのシステムで処理できるサーバは 100 未満です。

- **連続接続**　プライマリ・サイトとレプリケート・サイト間の接続に、広域ネットワークを使用する場合があります。しかし、Replication Server は、システム内のサーバ間にデータ交換用のほぼ連続的な接続パスがあるという想定で設計されています。
- **遅延時間：短**　遅延時間が短いということは、システムにおいて、あるデータベースにデータが入力されてからそのデータが各データベースにレプリケートされるまでのタイムラグが短いということです。Replication Server の場合、通常のレプリケーション・メッセージは、プライマリ・サイトに入力が行われる数秒の間に送信されます。
- **容量：大**　ほぼ連続的に接続が行われておりパフォーマンスに優れている場合、Replication Server は大容量のメッセージを処理できます。
- **異機種データベース**　Replication Server は、主要な DBMS をいくつかサポートしており、レプリケーション中にオブジェクト名のマッピングができます。このため、異機種データベースがサポートされます。

レプリケート・サイトとプライマリ・サイト

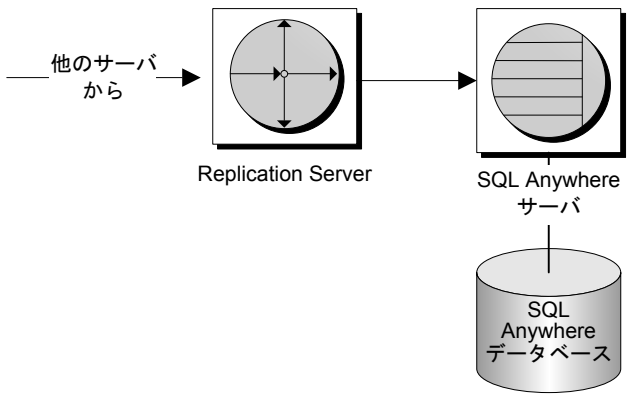
Replication Server のインストールでは、データベース間で共有されるデータは「**レプリケーション・サブスクリプション**」に配列されます。

各レプリケーション定義に対して「**プライマリ・サイト**」があり、ここでレプリケーション内のデータへの変更が行われます。レプリケーション内のデータを受信するサイトを「**レプリケート・サイト**」と呼びます。

レプリケート・サイトのコンポーネント

SQL Anywhere をレプリケート・サイトとして使用することができます。SQL Anywhere をレプリケート・サイトとして使用する場合、LTM は必要ありません。

次の図は、SQL Anywhere が Replication Server のインストール環境にレプリケート・サイトとして加わるために必要なコンポーネントを示しています。



- Replication Server がプライマリ・サイト・サーバからデータの変更を受信します。
- Replication Server が SQL Anywhere に接続して、変更を適用します。
- SQL Anywhere がデータベースに変更を加えます。

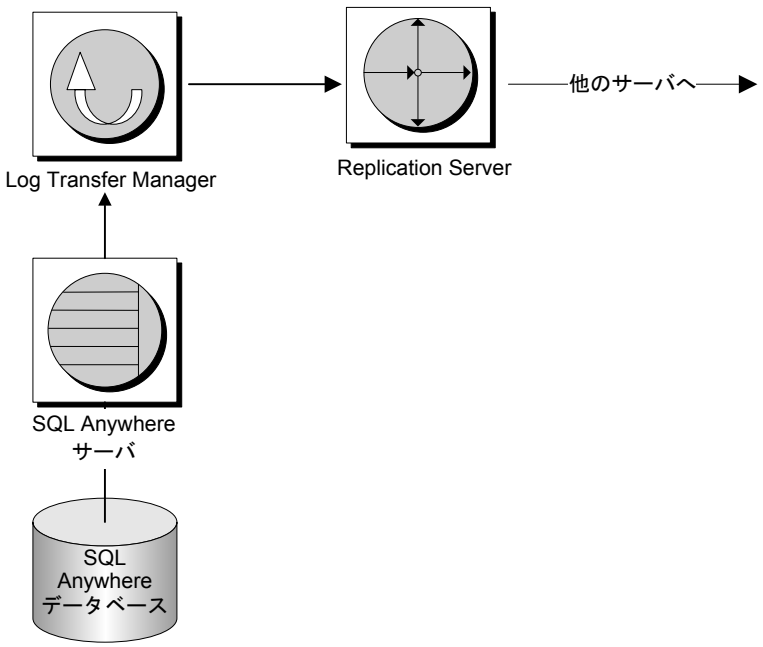
非同期プロシージャ・コール (APC)

Replication Server は、レプリケート・サイトで「**非同期プロシージャ・コール**」(APC) を使用して、プライマリ・サイト・データベースのデータを変更できます。APC を使用している場合は、前述の図は適用されません。ただし、稼働条件はプライマリ・サイトの場合と同じです。

プライマリ・サイトのコンポーネント

SQL Anywhere データベースをプライマリ・サイトとして使用するには、SQL Anywhere 用の Replication Agent である Log Transfer Manager (LTM) を使用する必要があります。LTM は Replication Server バージョン 10.0 以上をサポートします。

次の図は、SQL Anywhere が Replication Server のインストール環境にプライマリ・サイトとして加わるために必要なコンポーネントを示しています。図中の矢印はデータのフローを表しています。



- SQL Anywhere データベース・サーバがデータベースを管理します。
- SQL Anywhere の Log Transfer Manager がデータベースに接続します。トランザクション・ログをスキャンしてデータへの変更を取り出し、Replication Server に送信します。

- Replication Server はレプリケート・サイト・データベースに変更を送信します。

チュートリアル：Replication Server を使用したデータのレプリケート

この項は、プライマリ・データベースからレプリケート・データベースにデータをレプリケートする方法を、順を追って説明するチュートリアルです。扱うデータベースは、2 つとも SQL Anywhere データベースです。

Replication Server の前提

この項では、Replication Server がすでに稼働していることを前提としています。

Replication Server のインストールまたは設定の詳細については、Replication Server のマニュアルを参照してください。

チュートリアルの内容

このチュートリアルでは、テーブルのみをレプリケートする方法を説明します。

プロシージャのレプリケートについては、「レプリケーションのためのプロシージャと関数の準備」 1017 ページを参照してください。

このチュートリアルでは、(非常に) 初歩的なオフィス・ニュース・システムの簡単な例を使用します。それは、整数を 1 つ保有する ID カラム、ニュース・アイテムの作者の ID を保有するカラム、ニュース・アイテムのテキストを保有するカラムが 1 つずつある単一のテーブルです。id カラムと author カラムがプライマリ・キーを構成します。

チュートリアルで作成したファイルを保存するディレクトリ (たとえば、*c:¥tutorial*) を作成してから、チュートリアルでの作業を始めてください。

レッスン 1： SQL Anywhere データベースの作成

この項では、レプリケーション用に SQL Anywhere データベースを作成してセット・アップする方法について説明します。

データベースは、Sybase Central または dbinit ユーティリティを使用して作成できます。このチュートリアルでは、dbinit ユーティリティを使用します。

♦ **プライマリ・サイト・データベースを作成するには、次の手順に従います。**

1. 作成しておいたチュートリアル・ディレクトリ (*c:¥tutorial¥primedb* など) から次のコマンドを入力します。

   ```
   dbinit primedb
   ```

   現在のディレクトリにデータベース・ファイル *primedb.db* が作成されます。

♦ **レプリケート・サイト・データベースを作成するには、次の手順に従います。**

1. 作成しておいたチュートリアル・ディレクトリ (*c:¥tutorial¥repdb* など) から次のコマンドを入力します。

```
dbinit repdb
```

現在のディレクトリにデータベース・ファイル *repdb.db* が作成されます。

次の作業

次は、これらのデータベースを実行するデータベース・サーバを起動します。

レッスン 2： データベース・サーバの起動

プライマリ・データベースをロードした状態で、プライマリ・サイト・データベース・サーバを稼働させます。

♦ **プライマリ・サイト・データベース・サーバを起動するには、次の手順に従います。**

1. チュートリアル・ディレクトリに移動します。
2. 次のコマンドを入力して、primedb データベースを実行するネットワーク・データベース・サーバを起動します。デフォルトの通信ポート (2638) では TCP/IP ネットワーク通信プロトコルを使用してください。

```
dbsrv10 -x tcpip primedb.db
```

♦ **レプリケート・サイト・データベース・サーバを起動するには、次の手順に従います。**

1. チュートリアル・ディレクトリに移動します。
2. 次のコマンドを入力して、repdb データベースを実行するネットワーク・データベース・サーバを起動します。このサーバは、別のポートで実行してください。

```
dbsrv10 -x tcpip(PORT=2639) -n REPSV repdb.db
```

次の作業

次は、各 SQL Anywhere サーバへのエントリを interfaces ファイルに入れて、Replication Server がそれらのデータベース・サーバと通信できるようにします。

レッスン 3：システムへの Open Server のセット・アップ

システム内の Open Server のリストに一連の Open Server を追加する必要があります。

Open Server の追加

Open Server は、DSEdit ユーティリティを使用して interfaces ファイル (*SQL.ini*) 内に定義します。UNIX を使用している場合は、interfaces ファイルの名前は *interfaces*、ユーティリティの名前は sybinit です。

interfaces ファイルに定義を追加する方法については、「Open Server の設定」 990 ページを参照してください。

必要な Open Server

各 Open Server 定義に、「**名前**」と「**アドレス**」を 1 つずつ与えてください。定義のその他の属性は変更しないでください。次のそれぞれに Open Server エントリを追加する必要があります。

- **プライマリ・データベース**　次のアドレスを持つ PRIMEDB エントリを作成します。
  - **プロトコル**　NLWNSCK
  - **ネットワーク・アドレス**　localhost,2638
- **レプリケート・データベース**　次のアドレスを持つ REPDB エントリを作成します。
  - **プロトコル**　NLWNSCK
  - **ネットワーク・アドレス**　localhost,2639
- **プライマリ・データベースにある LTM**　これは、LTM を正常に停止させるために必要です。次のアドレスを持つ PRIMELTM というエントリを作成します。
  - **プロトコル**　NLWNSCK
  - **ネットワーク・アドレス**　localhost,2640
- **Replication Server**　このチュートリアルでは、Replication Server の Open Server は定義済みであることを前提とします。

次の作業

次は、Open Server が正しく設定されていることを確認します。

レッスン 4：Open Server が正しく設定されているかどうかの確認

DSEDIT ユーティリティから [ServerObject] − [Ping Server] を選択して、各 Open Server が使用可能であることを確認できます。

または、isql ユーティリティなどの Open Client アプリケーションを使用してデータベースに接続しても、各 Open Server が正しく設定されていることを確認できます。

プライマリ・サイト・データベース上で isql を実行するには、次のように入力します。

```
isql -U DBA -P sql -S PRIMEDB
```

Open Client の isql ユーティリティは、SQL Anywhere の Interactive SQL ユーティリティと同じではありません。

レッスン 5：プライマリ・データベースへの Replication Server 情報の追加

プライマリ・サイト・データベースを Replication Server インストール環境に加えるには、Replication Server のテーブルとプロシージャをそのデータベースに追加する必要があります。また、Replication Server が使用するユーザ ID を 2 つ作成してください。SQL Anywhere には SQL コマンド・ファイル *rssetup.sql* が付属しています。このファイルは次のタスクを実行します。

rssetup.sql コマンド・ファイルは、SQL Anywhere サーバで Interactive SQL ユーティリティから実行してください。

♦ **rssetup スクリプトを実行するには、次の手順に従います。**

1. Interactive SQL から DBA として SQL Anywhere データベースに接続します。
2. 次のコマンドを使用して rssetup スクリプトを実行します。

   ```
   read "install-dir¥scripts¥rssetup.sql"
   ```

   このスクリプトでは、*install-dir* に実際の SQL Anywhere インストール・ディレクトリを指定します。

   または、[ファイル] – [スクリプトの実行] を選択して、ファイルを検索します。

rssetup.sql によって実行されるアクション

rssetup.sql コマンド・ファイルは次の機能を実行します。

- パスワード **dbmaint** と DBA パーミッションを持つユーザ **dbmaint** を作成します。これは、プライマリ・サイト・データベースに接続するために Replication Server が必要とするメンテナンス・ユーザ名とパスワードです。
- パスワード **sysadmin** と DBA パーミッションを持つユーザ **sa** を作成します。これは、Replication Server がデータを表示するときに使用するユーザ ID です。
- **sa** と **dbmaint** を **rs_systabgroup** という名前のグループに追加します。

パスワードとユーザ ID

ハード・ワイヤされたユーザ ID (dbmaint と sa) とパスワードは、テストやチュートリアルには便利ですが、セキュリティを必要とするデータベースを実行するときには、パスワードだけでなくユーザ ID も変更してください。DBA パーミッションを付与されたユーザは、SQL Anywhere に対する完全な権限を持ちます。

ユーザ ID sa とそのパスワードは、Replication Server のシステム管理者アカウントのユーザ ID とパスワードと一致させてください。SQL Anywhere では、現在のところ NULL パスワードは使用できません。

パーミッション

rssetup.sql スクリプトは、パーミッション管理を一部含む多数のオペレーションを実行します。ここでは、*rssetup.sql* が行うパーミッション変更について説明します。*ユーザがパーミッションを変更する必要はありません。*

レプリケーションの際は、dbmaint ユーザと sa ユーザが所有者を明示的に指定しなくても、このテーブルにアクセスできることを確認してください。そのためには、テーブル所有者のユーザ ID にグループ・メンバシップ・パーミッションが必要であり、dbmaint ユーザと sa ユーザはテーブル所有者グループのメンバでなければなりません。グループ・パーミッションを付与するには、DBA 権限が必要です。

たとえば、ユーザ DBA がテーブルを所有している場合は、DBA にグループ・パーミッションを付与してください。


```
GRANT GROUP
TO DBA
```

次に、dbmaint ユーザと sa ユーザに DBA グループのメンバシップを付与してください。グループ・パーミッションを付与するには、DBA 権限またはグループ ID が必要です。

```
GRANT MEMBERSHIP
IN GROUP "DBA"
TO dbmaint ;
GRANT MEMBERSHIP
IN GROUP "DBA"
TO sa;
```

レッスン 6：プライマリ・データベース用のテーブルの作成

この項では、isql を使用してプライマリ・サイト・データベースにテーブルを 1 つ作成します。まず、プライマリ・サイト・データベースに接続されていることを確認します。

```
isql -U DBA -P sql -S PRIMEDB
```

次に、そのデータベースでテーブルを作成します。

```
CREATE TABLE news (
  ID INT,
  AUTHOR CHAR( 40 ) DEFAULT CURRENT USER,
  TEXT CHAR( 255 ),
  PRIMARY KEY ( ID, AUTHOR )
)
go
```

識別子の大文字と小文字の区別
SQL Anywhere では、すべての識別子で大文字と小文字が区別されません。Adaptive Server Enterprise では、デフォルトの場合、大文字と小文字の区別があります。Adaptive Server Enterprise との互換性を損なわないようにするため、SQL Anywhere でも識別子の大文字と小文字を区別して、SQL 文のあらゆる部分で一致させてください。
SQL Anywhere のパスワードでは、常に大文字と小文字が区別されます。ユーザ ID と識別子については、すべての SQL Anywhere データベースで大文字と小文字は区別されません。
詳細については、「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

news をレプリケーション・プライマリ・サイトの一部として機能させるためには、ALTER TABLE 文を使用するテーブルの REPLICATE フラグを ON に設定する必要があります。

```
ALTER TABLE news
REPLICATE ON
go
```

これは、Adaptive Server Enterprise のテーブルに対して sp_setreplicate または sp_setreptable プロシージャを実行することと同じです。REPLICATE ON は CREATE TABLE 文には設定できません。

レッスン 7：レプリケート・データベースへの Replication Server 情報の追加

プライマリ・データベースで実行したときとまったく同じ方法で、レプリケート・データベースで *rssetup.sql* コマンド・ファイルを実行してください。また、dbmaint ユーザと sa ユーザがテーブル所有者を明示的に指定しなくても、このテーブルにアクセスできることを確認してください。

以上の作業は、プライマリ・データベースで実行した作業と同じです。

詳細については、「レッスン 5：プライマリ・データベースへの Replication Server 情報の追加」 1006 ページを参照してください。

レッスン 8：レプリケート・データベース用のテーブルの作成

レプリケート・サイト・データベースには、受信するデータを保持するテーブルが必要です。ここで、このためのテーブルを作成します。Replication Server のインストール環境では、データベースの要素が正しい場所に用意されているかぎり、特別な文がなくてもその要素はレプリケート・サイトとして機能します。特に、REPLICATE を ON に設定する必要はありません。これはプライマリ・サイトでのみ必要です。

Replication Server は、名前が異なるテーブルとカラムの間のレプリケーションを可能にします。しかし、簡単な例として、レプリケート・データベースに、プライマリ・データベースのテーブルと定義が同一であるテーブルを作成します (ただし、レプリケート・データベースでは REPLICATE を ON に設定しない点を除く)。これを作成する文は、次のとおりです。

```
CREATE TABLE news (
  ID INT,
  AUTHOR CHAR( 40 ) DEFAULT CURRENT USER,
  TEXT CHAR( 255 ),
  PRIMARY KEY ( ID, AUTHOR )
)
go
```

チュートリアルでは、CREATE TABLE 文をプライマリ・サイトの CREATE TABLE 文と*厳密に*同じものにしてください。

dbmaint ユーザと sa ユーザが所有者名を指定しなくてもこのテーブルにアクセスできることを確認してください。また、これらのユーザ ID には、テーブルに対する SELECT パーミッションと UPDATE パーミッションが必要です。

レッスン 9：Replication Server の設定

Replication Server では、次のタスクを実行する必要があります。

- プライマリ・サイト・データ・サーバの接続を作成する。
- レプリケート・サイト・データ・サーバの接続を作成する。
- レプリケーション定義を作成する。
- レプリケーションへのサブスクリプションを作成する。

- SQL Anywhere LTM を起動する。

プライマリ・サイトの接続の作成

isql を使用して Replication Server に接続し、プライマリ・サイトの SQL Anywhere データベースへの接続を作成します。

次のコマンドは、PRIMEDB Open Server に primedb データベースへの接続を作成します。

```
CREATE CONNECTION TO PRIMEDB.primedb
SET ERROR CLASS rs_sqlserver_error_class
SET FUNCTION STRING class rs_sqlserver_function_class
SET USERNAME dbmaint
SET PASSWORD dbmaint
WITH LOG TRANSFER ON
go
```

rssetup.sql コマンド・ファイルで dbmaint ユーザ ID とパスワードを変更した場合は、このコマンドの dbmaint ユーザ名とパスワードを置き換えてください。

Replication Server は実際にデータベース名 primedb を使用するわけではありません。データベース名は、PRIMEDB Open Server のコマンド・ラインから読み込まれます。しかし、構文を一致させるために、CREATE CONNECTION 文にデータベース名を入れてください。

接続を作成する文の詳細については、*『Replication Server リファレンス・マニュアル』* の「Replication Server コマンド」の章を参照してください。

レプリケート・サイトの接続の作成

isql を使用して Replication Server に接続し、レプリケート・サイトの SQL Anywhere データベースへの接続を作成します。

次のコマンドは、REPDB Open Server に repdb データベースへの接続を作成します。

```
CREATE CONNECTION TO REPDB.repdb
SET ERROR CLASS rs_sqlserver_error_class
SET FUNCTION STRING CLASS rs_sqlserver_function_class
SET USERNAME dbmaint
SET PASSWORD dbmaint
go
```

この文は、WITH LOG TRANSFER ON 句がないプライマリ・サイト・サーバ用の文とは違います。

rssetup.sql コマンド・ファイルで dbmaint ユーザ ID とパスワードを変更した場合は、このコマンドの dbmaint ユーザ名とパスワードを置き換えてください。

レプリケーション定義の作成

isql を使用して Replication Server に接続し、レプリケーション定義を作成します。次の文は、primedb データベースにニュース・テーブルについてのレプリケーション定義を作成します。


```
CREATE REPLICATION DEFINITION NEWS
WITH PRIMARY AT PRIMEDB.primedb
( id INT, author CHAR(40), text CHAR(255) )
PRIMARY KEY ( id, author )
go
```

CREATE REPLICATION DEFINITION 文の詳細については、*『Replication Server リファレンス・マニュアル』* を参照してください。

LTM 設定ファイルで qualify_table_owners オプションを On に設定した場合は、レプリケートするすべてのテーブルのテーブル所有者を文の中に指定する必要があります。

SQL Anywhere LTM の設定と起動

レプリケーションを実行するには、SQL Anywhere LTM がプライマリ・サイト・サーバに対して実行されていなければなりません。LTM 設定ファイルを編集して SQL Anywhere LTM を正しく設定してから、SQL Anywhere LTM を起動してください。

次は、primedb データベース用の設定ファイルの例です。例にならう場合は、このファイルのコピーを *primeltm.cfg* として作成してください。

```
#
# Configuration file for 'PRIMELTM'
#
SQL_server=PRIMEDB
SQL_database=primedb
SQL_user=sa
SQL_pw=sysadmin
RS_source_ds=PRIMEDB
RS_source_db=primedb
RS=your_rep_server_name_here
RS_user=sa
RS_pw=sysadmin
LTM_admin_user=DBA
LTM_admin_pw=sql
LTM_charset=cp850
scan_retry=2
APC_user=sa
APC_pw=sysadmin
SQL_log_files=C:¥TUTORIAL
```

rssetup.sql コマンド・ファイルでユーザ ID とパスワードを sa と sysadmin から変更した場合は、この設定では新しいユーザ ID とパスワードを使用してください。

プライマリ・サイト・サーバで稼働する SQL Anywhere LTM を起動するには、次のコマンドを使用します。

```
dbltm -S PRIMELTM -C primeltm.cfg
```

接続情報は *primeltm.cfg* に保存されています。このコマンドでは、LTM のサーバ名は PRIMELTM です。

次の文を入力すると、SQL Anywhere LTM についての使用情報を検索できます。

```
dbltm -?
```

SQL Anywhere LTM は、Windows サービスとして実行させることができます。

サービスとしてのプログラムの実行については、「現在のセッション外でのサーバの起動」 36 ページを参照してください。

レプリケーションのためのサブスクリプションの作成

isql を使用して Replication Server に接続し、レプリケーションのためのサブスクリプションを作成します。

次の文は、レプリケート・サイトを repdb データベースとして、ニュース・レプリケーションのためのサブスクリプションを作成します。

ニュース・レプリケーションの詳細については、「レプリケーション定義の作成」 1010 ページを参照してください。

```
CREATE SUBSCRIPTION NEWS_SUBSCRIPTION
FOR news
WITH REPLICATE AT REPDB.repdb
go
```

これでインストールは終了しました。データをレプリケートして、セット・アップが正しく機能しているかどうか確認してください。

レッスン 10：プライマリ・サイトでのレプリケーション用データの入力

プライマリ・データベースからレプリケート・データベースにデータをレプリケートできるようになりました。例として、isql ユーティリティを使用してプライマリ・データベースに接続し、news テーブルにローを 1 つ入力します。

```
INSERT news (id, text)
VALUES (1, 'Test news item.' )
COMMIT
go
```

SQL Anywhere LTM は Replication Server に対し、コミットされた変更しか送信しません。データの変更は、次回、LTM がトランザクション・ログをポーリングするときにレプリケートされます。

チュートリアル終了

これでチュートリアルは終了しました。

Replication Server のデータベースの設定

Replication Server のインストール環境に加わる SQL Anywhere データベースは、あらかじめ個別に設定しておく必要があります。データベースの設定には、次のタスクが含まれます。

- メンテナンス・ユーザのセキュア・ユーザ ID と、データを表示するときに Replication Server が使用する名前の選択
- Replication Server 用のデータベースの設定
- 言語と文字セットの設定 (必要に応じて)

LTM の設定

Replication Server にデータを送信するには、プライマリ・サイトの SQL Anywhere データベースごとに LTM が 1 つずつ必要です。Replication Server をデータベースに接続するには、プライマリ・サイトまたはレプリケート・サイトの SQL Anywhere データベースごとに Open Server 定義が 1 つずつ必要です。

コンジットの設定については、「LTM の設定」 1019 ページを参照してください。

Replication Server のデータベースの設定

SQL Anywhere データベースと、データベース内の必要なテーブルなどを作成したら、データベースを Replication Server で使用できるように準備してください。それには、SQL Anywhere Replication Agent 製品に付属の設定スクリプトを使用します。このスクリプトの名前は、*rssetup.sql* です。

設定スクリプトを実行する必要があるとき

Replication Server インストール環境に加わる SQL Anywhere データベースは、プライマリ・サイトまたはレプリケート・サイトのどちらとして加わるかにかかわらず、必ず設定スクリプトを実行する必要があります。

設定スクリプトの機能

設定スクリプトは、データベースへの接続時に Replication Server に必要なユーザ ID を作成します。また、Replication Server が使用する一連のストアド・プロシージャとテーブルも作成します。テーブルは **rs_** という文字で始まり、プロシージャは **sp_** という文字で始まります。プロシージャには、文字セットと言語を設定するのに重要なものがあります。

設定スクリプトを実行する準備

Replication Server は、テーブルがレプリケートされた各ローカル・データベースに対して、特殊なデータ・サーバ、「**メンテナンス・ユーザ**」ログイン名を使用します。これによって、Replication Server はデータベース内のレプリケートされたテーブルを維持、更新できます。

メンテナンス・ユーザ

設定スクリプトは、名前が **dbmaint** でパスワードが **dbmaint** のメンテナンス・ユーザを作成します。メンテナンス・ユーザには、SQL Anywhere データベースの DBA パーミッションがあるので、データベースを完全に制御できます。セキュリティのため、メンテナンス・ユーザの ID とパスワードを変更してください。

♦ **メンテナンス・ユーザの ID とパスワードを変更するには、次の手順に従います。**

1. テキスト・エディタで *rssetup.sql* 設定スクリプトを開きます。このスクリプトは、SQL Anywhere のインストール・ディレクトリの *scripts* サブディレクトリにあります。
2. dbmaint ユーザ ID のすべてのオカレンスを、新しいメンテナンス・ユーザ ID に変更します。
3. dbmaint パスワードを新しいメンテナンス・ユーザ・パスワードに変更します。そのパスワードは、設定スクリプト・ファイル上部の次の場所に表示されます。

```
GRANT CONNECT TO dbmaint
IDENTIFIED BY dbmaint;
```

メンテナンス・ユーザ ID

Replication Server は、データベースに接続してレプリケーション内のデータの最初のコピーを表示するときに、Replication Server システム管理者アカウントを使用して実行します。

Replication Server システム管理者のユーザ ID とパスワードと、SQL Anywhere データベースにあるユーザ ID とパスワードが一致する必要があります。SQL Anywhere では NULL パスワードは使用できません。

この設定スクリプトは、Replication Server 管理者がユーザ ID sa とパスワード **sysadmin** を持っていることを前提としています。これを変更して、実際の名前とパスワードに一致させてください。

♦ **システム管理者のユーザ ID とパスワードを変更するには、次の手順に従います。**

1. テキスト・エディタで *rssetup.sql* 設定スクリプトを開きます。
2. ユーザ ID sa が記述されているすべての部分を、Replication Server システム管理者のユーザ ID と一致するように変更します。
3. パスワード sa を変更して、Replication Server システム管理者のパスワードと一致させます。

   このパスワードには、**sysadmin** という初期設定値があります。

設定スクリプトの実行

ユーザ ID とパスワードがそれぞれ一致するように設定スクリプトを修正したら、設定スクリプトを実行して、SQL Anywhere データベースにメンテナンス・ユーザとシステム管理者ユーザを作成できます。

♦ **設定スクリプトを実行するには、次の手順に従います。**

1. SQL Anywhere データベース・サーバで SQL Anywhere データベースを起動します。
2. Interactive SQL ユーティリティを起動し、DBA としてデータベースに接続します。

   作成した SQL Anywhere データベースには、パスワードが **sql** であるユーザ ID **DBA** が定義されています。このユーザには DBA 権限があります。
3. [SQL 文] ウィンドウ枠に次のコマンドを入力して、スクリプトを実行します。

   ```
   read install-dir¥scripts¥rssetup.sql
   ```

   このコマンドでは、*install-dir* に実際の SQL Anywhere インストール・ディレクトリを指定します。

Replication Server の文字セットと言語について

SQL Anywhere データベースが作成されると、特定の照合 (文字セットとソート順) が割り当てられます。Replication Server は、文字セットとソート順に異なる識別子セットを使用します。

LTM 設定ファイルで文字セットと言語のパラメータを設定します。指定する文字セット・ラベルがわからない場合は、次の方法でサーバの文字セットを確認してください。

♦ **文字セットを確認するには、次の手順に従います。**

· 次のコマンドを実行します。

```
exec sp_serverinfo csname
```

言語ラベルのリストについては、「言語ラベルの値」 347 ページを参照してください。

Replication Server の識別子

SQL Anywhere Replication Agent を Replication Server 15.0 と Open Client/Open Server 15.0 で使用する場合、Replication Agent は、テーブル名、カラム名、プロシージャ名、関数名、パラメータ名について、最大 128 バイトの長さをサポートします。

Replication Agent を Replication Server と Open Client/Open Server の以前のバージョンで使用する場合は、識別子の最大長は 30 バイトとなります。

LTM の使用

SQL Anywhere LTM は SQL Anywhere トランザクション・ログ内の情報に依存しているので、バックアップを保存せずにそのログを削除したり損傷したりしないように気をつけてください。バックアップの保存には、トランザクション・ログ・ミラーなどを使用します。

トランザクション・ログの管理については、「トランザクション・ログとバックアップの管理」 1022 ページを参照してください。

SQL Anywhere LTM を Adaptive Server Enterprise LTM の代わりとして使用することはできません。トランザクション・ログのフォーマットが異なるからです。

SQL Anywhere LTM は、Transact-SQL ダイアレクトのストアド・プロシージャが呼び出したレプリケーションだけではなく、挿入、更新、削除のレプリケーションもサポートします。

Adaptive Server Enterprise LTM では、Replication Server にデータ変更を送信してから、データ変更がコミットされます。Replication Server は、COMMIT 文を受信するまで変更を保持します。一方、SQL Anywhere LTM では、コミットされた変更だけを Replication Server に送信します。長いトランザクションの場合は、そのことがレプリケーションの遅れの原因になることがあります。すべての変更が、Replication Server を介してから分散されるからです。

レプリケーションのテーブルの設定

sp_setreplicate システム・プロシージャ、sp_setrepproc システム・プロシージャ、または ALTER TABLE 文を使用して、レプリケーションのテーブルを設定できます。テーブルは、単一の句の ALTER TABLE 文を使用し、プライマリ・データ・ソースとして識別されます。

```
ALTER TABLE table-name
SET REPLICATE ON;
```

テーブルの REPLICATE ON 設定が及ぼす影響

REPLICATE ON を設定すると、トランザクション・ログに追加の情報が入ります。テーブルで UPDATE、INSERT、または DELETE アクションがあったときは必ず入ります。この追加の情報は、必要に応じて、SQL Anywhere Replication Agent がローの完全な更新前イメージをレプリケーション用に Replication Server へ送信するときに使用されます。

テーブル中のデータの一部だけをレプリケートする必要がある場合でも、テーブルに対する変更のすべてが Replication Server に送信されます。レプリケートするデータとレプリケートしなくてよいデータを区別するのは、Replication Server です。

1 つのローを更新、挿入または削除する場合、ローの更新前イメージがそのアクションの前のローの内容であり、更新後イメージがアクションの後のローの内容となります。INSERT の場合は、更新後イメージだけが送信されます (更新前イメージは空です)。DELETE の場合は、更新後イメージが空で、更新前イメージだけ送信されます。UPDATE の場合は、更新前イメージと更新されたイメージの両方が送信されます。

次のデータ型は、レプリケーション用にサポートされます。

データ型	説明 (Open Client/Open Server タイプ)
Exact integer データ型	int、smallint、tinyint
Exact decimal データ型	decimal、numeric
Approximate numeric データ型	float (8 バイト)、real
Money データ型	money、smallmoney
文字データ型	char(n)、varchar(n)、text
日付と時刻データ型	datetime、smalldatetime
バイナリ・データ型	binary(n)、varbinary(n)、image
Bit データ型	bit

注意

SQL Anywhere は、NULL ではない、長さ 0 のデータをサポートします。ただし、長さが 0 の null 以外の varchar と binary データは、レプリケート・サイトに NULL としてレプリケートされます。

プライマリ・テーブルの中にサポートされないデータ型のカラムがある場合は、互換性があり、サポートされているデータ型を使用してレプリケーション定義を作成すれば、そのデータをレプリケートできます。たとえば、DOUBLE カラムをレプリケートするには、レプリケーション定義の中でそれを FLOAT と定義します。

テーブルの REPLICATE ON 設定が及ぼすその他の影響

大量に更新されたテーブルでは、レプリケーション・パフォーマンスが重大な影響を受けることがあります。レプリケーション・トラフィックに関連するパフォーマンス問題が発生したら、複写プロシージャの使用を考えてください。複写プロシージャは、個別のアクションではなく、プロシージャに対する呼び出しだけを送信するからです。

REPLICATE ON を設定すると、追加の情報がトランザクション・ログに送信されるので、このログはレプリケートしないデータベースの場合より早く大きくなります。

最少カラムのレプリケーション定義

SQL Anywhere LTM は、Replication Server のレプリケート最少カラム機能をサポートします。この機能は、Replication Server で有効になります。

レプリケート最少カラムの詳細については、Replication Server のマニュアルを参照してください。

レプリケーションのためのプロシージャと関数の準備

ストアド・プロシージャを使用してテーブルのデータを修正できます。更新、挿入、削除は、プロシージャ内から実行されます。

Replication Server は、プロシージャが特定の条件を満たす場合、そのプロシージャをレプリケートできます。プロシージャ内の最初の文は、プロシージャがレプリケートされるように更新しなければなりません。

Replication Server がプロシージャをレプリケートする方法の詳細については、Replication Server のマニュアルを参照してください。

SQL Anywhere は、ストアド・プロシージャに対して、2 種類のダイアレクトをサポートします。ISO/ANSI 規格案に基づく Watcom-SQL ダイアレクトと、Transact-SQL ダイアレクトです。レプリケーションのためのストアド・プロシージャを記述するときには、Transact-SQL を使用してください。

関数 APC フォーマット

SQL Anywhere LTM は、Replication Server の「**関数 APC**」フォーマットをサポートします。これらの関数を使用するには、設定パラメータの **rep_func** を **on** (デフォルトは **off**) に設定します。

LTM は、レプリケートされたすべての APC を、テーブル APC または関数 APC のいずれかに解釈します。単一の SQL Anywhere データベースでは、関数 APC を他のテーブル APC と組み合わせることはできません。

レプリケート関数の詳細については、Replication Server のマニュアルを参照してください。

プロシージャ・レプリケーションを制御するための SQL 文

プロシージャを、ALTER PROCEDURE 文を使用して、レプリケーション・ソースとして機能するように設定できます。

次の文は、プロシージャ MyProc をレプリケーション・ソースとして機能させます。

```
ALTER PROCEDURE MyProc
REPLICATE ON;
```

次の文は、プロシージャ MyProc がレプリケーション・ソースとして機能するのを防ぎます。

```
ALTER PROCEDURE MyProc
REPLICATE OFF;
```

sp_setreplicate または sp_setrepproc システム・プロシージャを使用して、レプリケーションのためのプロシージャを設定することもできます。

プロシージャのための REPLICATE ON 設定が及ぼす影響

プロシージャがレプリケーション・データ・ソースとして使用される場合は、プロシージャを呼び出すと追加の情報がトランザクション・ログに送られます。

非同期プロシージャ

プライマリ・サイト・データベースでデータを更新するためにレプリケート・サイトで呼び出されるプロシージャが「**非同期プロシージャ**」です。このプロシージャはレプリケート・サイトではアクションを実行しませんが、このプロシージャに対する呼び出しがプライマリ・サイトへレ

プリケートされ、そこで同じ名前のプロシージャが実行されます。これを「**非同期プロシージャ・コール**」 (APC) と呼びます。それから、APC が加えた変更が、プライマリ・データベースからレプリケート・データベースに通常の方法でレプリケートされます。

APC については、Replication Server のマニュアルを参照してください。

APC_user と APC サポート

SQL Anywhere の APC に対するサポートは、Adaptive Server Enterprise でのサポートとは異なります。Adaptive Server Enterprise では、それぞれの APC が、レプリケート・サイトでプロシージャを呼び出したユーザのユーザ ID とパスワードを使用して実行されます。一方、SQL Anywhere では、パスワードをトランザクション・ログに格納しないので、プライマリ・サイトでは使用できません。この違いを回避するために、対応するパスワードがある単一のユーザ ID を LTM 設定ファイルに入力し、プライマリ・サイトでこのユーザ ID (**APC_user**) を使用してプロシージャを実行します。したがって、APC_user は、呼び出される可能性があるそれぞれの APC に対する適切なパーミッションを、プライマリ・サイトで付与されていなければなりません。

LTM の設定

LTM の動作は、LTM の「**設定ファイル**」を修正することにより制御します。このファイルは、テキスト・エディタを使用して作成、編集できる一般的なテキスト・ファイルです。LTM 設定ファイルには、LTM に必要な情報が含まれています。たとえば、LTM がログをどの SQL Anywhere サーバから転送しているか、どの Replication Server に転送するかなどの情報です。LTM を実行するには、有効な設定ファイルが必要です。

設定ファイルの作成

テキスト・エディタを使用して設定ファイルを作成してから、LTM を実行してください。-C LTM コマンドは、使用する設定ファイルの名前を指定します。デフォルトは *dbltm.cfg* です。

設定ファイルのフォーマット

LTM 設定ファイルのフォーマットは、*『Replication Server 管理ガイド』* で説明されている Replication Server の設定ファイルのフォーマットと同じです。その概要は次のとおりです。

- 設定ファイルでは、各行にエントリが 1 つずつ含まれています。
- 1 つのエントリは、パラメータ 1 つと、その後に続く = 文字と、さらにその後に続く値とで構成されています。

  ```
  Entry=value
  ```

- # 文字で始まる行はコメントであり、LTM はこれを無視します。
- 設定ファイルには先行ブランクを含めません。
- エントリは、大文字と小文字が区別されます。

使用できる設定ファイル・パラメータの完全なリストについては、「LTM 設定ファイル」 684 ページを参照してください。

設定ファイルの例

- 次に、SQL Anywhere LTM 設定ファイルの例を示します。

```
# This is a comment line
# Names are case sensitive.
SQL_user=sa
SQL_pw=sysadmin
SQL_server=PRIMESV
SQL_database=primedb
RS_source_ds=PRIMESV
RS_source_db=primedb
RS=MY_REPSERVER
RS_user=sa
RS_pw=sysadmin
LTM_admin_user=DBA
LTM_admin_pw=sql
LTM_charset=cp850
scan_retry=2
SQL_log_files=e:¥logs¥backup
APC_user=sa
APC_pw=sysadmin
```

バッチによるトランザクションのレプリケーション

トランザクションのバッファリングによる影響

LTM は、Replication Server に対するレプリケーション・コマンドのバッファリングを可能にします。レプリケーション・コマンドをバッファし、それをバッチにして送信すると、送信されるメッセージの数が少なくなります。特に、大きなボリュームのインストールでは、総スループットが著しく増加することがあります。

バッチ・モードの機能

デフォルトで、LTM はトランザクションをバッファします。次の場合は、バッファがフラッシュします (トランザクションが Replication Server に送信されます)。

- **最大コマンド数に到達** **batch_ltl_sz** パラメータは、バッファに保存される最大 LTL (ログ転送言語) コマンド数を設定してから、フラッシュします。デフォルト設定値は 200 です。
- **最大使用メモリ量に到達** **batch_ltl_mem** パラメータは、バッファが占有できる最大メモリ量を設定してから、フラッシュします。デフォルト設定は 256 KB です。
- **トランザクション・ログ処理の完了** トランザクション・ログ内に処理するエントリがなくなると (つまり、LTM が、コミットされたすべてのトランザクションを処理した場合)、バッファはフラッシュします。

バッファリングの停止

batch_ltl_cmds パラメータを **off** にすることによって、トランザクションのバッファリングを停止できます。

```
batch_ltl_cmds=off
```

言語と文字セットの問題

言語と文字セットは、多くのレプリケーション・サイトで重要な問題です。システム内のデータベースとサーバは、特定の照合 (文字セットとソート順) を使用して文字列の保存と並べ替えを行います。SQL Anywhere の文字セット・サポートは、Adaptive Server Enterprise とその他の Open Client/Open Server ベースのアプリケーションにおける文字セット・サポートとは異なる方法で行われています。

この項では、SQL Anywhere データベースのデータを Replication Server と共有できるようにし、それによって他のデータベースとも共有できるように SQL Anywhere LTM を設定する方法を説明します。

LTM は、デフォルトの Open Client/Open Server の言語、ソート順、文字セットを自動的に使用します。これらのデフォルト値は、LTM 設定ファイルにエントリを追加することによって無効にできます。

Open Client/Open Server 照合

Adaptive Server Enterprise、Replication Server、その他の Open Client/Open Server のアプリケーションには、文字セットを管理する共通の手段があります。

Open Client/Open Server の文字セット・サポートについては、*『Adaptive Server Enterprise システム管理ガイド』* の「文字セット、ソート順、言語の設定」の章を参照してください。

Replication Server の文字セット問題の詳細については、*『Replication Server デザイン・ガイド』* の「国際的な複写システムの設計」の章を参照してください。

この項では、Open Client/Open Server の文字セット・サポートの概要を説明します。

国際化ファイル

特定の言語でのデータ処理をサポートするファイルは、「**国際化ファイル**」と呼ばれています。Adaptive Server Enterprise とその他の Open Client/Open Server のアプリケーションには、複数のタイプの国際化ファイルがあります。

使用している Sybase ディレクトリ内に、*charsets* というディレクトリがあります。*Charsets* にはサブディレクトリのセットがあります。サブディレクトリには、使用できる文字セットが 1 つずつ格納されています。各文字セットには、ファイルのセットが格納されています。次の表は、各ファイルを示しています。

ファイル	説明
Charset.loc	文字セット定義ファイルで、英数字、句読点、オペランド、大文字または小文字など、文字の表記プロパティを定義する。
**.srt*	英数字と特殊文字のためのソート順を定義する。
**.xlt*	ユーティリティとともに使用する、ターミナル特有の文字翻訳ファイル。

LTM 設定ファイル中の文字セット設定値

LTM 設定ファイルには、文字セット問題に関連する設定値が 3 つあります。

- **LTM_charset**　LTM で使用する文字セット。Sybase がサポートするすべての文字セットを指定できます。
- **LTM_language**　LTM がエラー・ログとクライアントにメッセージを出力するために使用する「**言語**」。LTM がローカライズされている言語で、LTM 文字セットと互換性のあるすべての言語を指定できます。

  SQL Anywhere LTM は、いくつかの言語にローカライズされています。

注意

文字セット　Open Client/Open Server 環境では、データ・サーバとそれに接続されている Replication Server と同じ文字セットを LTM でも使用してください。

SQL Anywhere の文字セットは Open Client/Open Server の文字セットとは別の指定をします。したがって、SQL Anywhere の文字セットが LTM の文字セットと互換性があることが必要です。

言語　Sybase リリース・ディレクトリの *locales* サブディレクトリにある *locales.dat* ファイルに、有効なマップ設定が含まれています。ただし、現在はユーザ・インタフェースの LTM 出力メッセージは LTM がローカライズされている言語で表示されます。

ソート順　レプリケーション・システムのすべてのソート順を同一にしてください。Sybase リリース・ディレクトリの *locales* サブディレクトリに保存されている *locales.dat* ファイルに、ユーザのプラットフォーム用のデフォルト・エントリがあります。

例

- 次の設定値は、日本語のインストールの場合に有効です。

```
LTM_charset=SJIS
LTM_language=Japanese
```

トランザクション・ログとバックアップの管理

Adaptive Server Enterprise LTM と SQL Anywhere LTM が異なる点として、Adaptive Server Enterprise LTM がテンポラリ・リカバリ・データベースに依存して古いトランザクションにアクセスするのに対し、SQL Anywhere LTM は古いトランザクション・ログへのアクセスに依存していることがあります。SQL Anywhere LTM には、テンポラリ・リカバリ・データベースはありません。

レプリケーションはトランザクション・ログ内のオペレーションへのアクセスに依存するので、SQL Anywhere のプライマリ・サイト・データベースの場合は、古いトランザクション・ログにアクセスする場合があります。この項では、SQL Anywhere のプライマリ・サイトでバックアップ・プロシージャを設定して、古いトランザクション・ログに正しくアクセスする方法を説明します。

トランザクション・ログ喪失の影響

SQL Anywhere のプライマリ・データベース・サイトでは、しっかりバックアップを取ることが重要です。トランザクション・ログを失うと、レプリケート・サイト・データベースを実体化し直さなければならないことになります。プライマリ・データベース・サイトでは、トランザクション・ログ・ミラーを使うことをおすすめします。

トランザクション・ログ・ミラーとバックアップ・プロシージャの詳細については、「バックアップとデータ・リカバリ」 811 ページを参照してください。

LTM 設定ファイルには、バックアップされたトランザクション・ログがどこに保存されているかを示すディレクトリ・エントリが含まれています。この項では、このディレクトリが適切な形のままであるようにバックアップ・プロシージャを設定する方法を説明します。

バックアップ・ユーティリティのオプション

バックアップ・ユーティリティには、バックアップ時にトランザクション・ログの名前を変更して再起動するオプションがあります。dbbackup ユーティリティでは、これが -r オプションです。プライマリ・データベースとレプリケーション・データベースのトランザクション・ログをバックアップするときには、このオプションを使用することをおすすめします。

たとえば、データベース *primedb.db* がディレクトリ *c:¥prime* の中にあり、トランザクション・ログがディレクトリ *d:¥primelog¥primedb.log* の中にあるとします。名前の変更と再起動のオプションを使用して、このトランザクション・ログをディレクトリ *e:¥primebak* にバックアップするには、次のタスクを実行します。

1. トランザクション・ログをバックアップして、バックアップ・ファイル *e:¥primebak¥primedb.log* を作成します。
2. 既存のトランザクション・ログの名前を *d:¥primelog¥YYMMDDxx.log* に変更します。**xx** は、**AA** ー **ZZ** のアルファベット順の英字です。
3. 新しいトランザクション・ログを *d:¥primelog¥primedb.log* として開始します。

バックアップを何回か行うと、ディレクトリ *d:¥primelog* に連続した名前の一連のトランザクション・ログができます。ログ・ディレクトリには、このバックアップ・プロシージャで生成された一連のログ以外のトランザクション・ログが含まれないようにしてください。

delete_old_logs オプションの使用

SQL Anywhere データベースの delete_old_logs オプションは、デフォルトで Off に設定されます。このデフォルトを On に設定すると、Replication Server がトランザクションにアクセスする必要がなくなったときに、LTM が自動的に古いトランザクション・ログを削除します。このオプションでは、レプリケーションの設定でディスク領域の管理がしやすくなることがあります。

たとえば、delete_old_logs オプションを PUBLIC グループに設定します。

```
SET OPTION PUBLIC.delete_old_logs = 'ON'
```

または

```
SET OPTION PUBLIC.delete_old_logs = '10 days'
```


詳細については、「delete_old_logs オプション [Mobile Link クライアント] [SQL Remote] [Replication Agent]」 447 ページを参照してください。

アンロード・ユーティリティとレプリケーション

データベースがレプリケーションに参加している場合は、データベースを実体化し直すことにならないように、アンロードと再ロードには注意が必要です。レプリケーションは、トランザクション・ログに基づいて行われます。データベースをアンロードして再ロードすると、古いトランザクション・ログが削除されます。このため、レプリケーションが関係するときは、バックアップをきちんと実行することが特に重要です。

データベース全体のレプリケーション

SQL Anywhere では、ショートカットを使ってデータベース全体をレプリケートするので、データベース内の各テーブルをレプリケートされるテーブルとして設定する必要はありません。

replicate_all と呼ばれる PUBLIC データベース・オプションを、SET OPTION 文を使用して設定できます。レプリケートするデータベース全体を、次のコマンドを使用して指定できます。

```
SET OPTION PUBLIC.replicate_all='On';
```

この設定とその他の PUBLIC オプションの設定値を変更するには、DBA 権限が必要です。新しい設定値を有効にするため、データベースを再起動してください。replicate_all オプションは、プロシージャには影響を及ぼしません。「replicate_all オプション [Replication Agent]」 489 ページを参照してください。

LTM の停止

LTM は、Windows ではユーザ・インタフェースから停止でき、それ以外の環境ではコマンドを発行して停止できます。

♦ **Windows でサービスとして稼働していない LTM を停止するには、次の手順に従います。**

- ユーザ・インタフェース上の [シャットダウン] をクリックします。

♦ **コマンドを発行して LTM を停止するには、次の手順に従います。**

1. LTM 設定ファイル内の LTM_admin_user ログイン名とパスワードを使用して、isql から LTM に接続します。ユーザ ID とパスワードは大文字と小文字を区別します。
2. SHUTDOWN 文を使用して LTM を停止します。

例

次の文は、isql を LTM PRIMELTM に接続して停止します。

```
isql -SPRIMELTM -UDBA -Psql
1> shutdown
2> go
```


パート VIII. SQLAnywhere for Windows CE

パート VIII では、Windows CE 上で SQL Anywhere を使用する方法について説明します。

第 27 章

SQL Anywhere for Windows CE

目次

Windows CE デバイスへの SQL Anywhere のインストール

前提条件

- Microsoft ActiveSync 3.5 以降
- SQL Anywhere がサポートしている Windows CE デバイス

  SQL Anywhere でサポートされている Windows CE デバイスのリストについては、「SQL Anywhere がサポートするプラットフォームおよびエンジニアリング・サポート状況」を参照してください。
- サポートされている Windows オペレーティング・システムを実行しているコンピュータ

Windows CE ファイルのロケーション

Windows CE 上での SQL Anywhere インストールのロケーションは、データベースのタイプとインストール先のロケーションによって異なります。サブディレクトリは作成されません。すべての DLL は、*¥Windows* ディレクトリにインストールされます。

オペレーティング・システム	ロケーション	インストール・ディレクトリ
Windows CE 4.x	メイン・メモリ	*¥Program Files¥SQLAny10*
Windows CE 4.x	ストレージ・カードまたは SD カード	*¥storage-card¥Sybase SQL Anywhere 10* *¥SD-card¥Sybase SQL Anywhere 10*
Windows CE 5.0	メイン・メモリ	*¥Program Files¥SQLAny10*
Windows CE 5.0	ストレージ・カードまたは SD カード	*¥storage-card¥SQLAny10* *¥SD-card¥SQLAny10*

インストール時の考慮事項： Windows CE での ICU の使用

ユニコード照合アルゴリズム (UCA) は、ユニコード文字セット全体のソートに使用するアルゴリズムです。これにより、言語的に正しい比較、ソート、大文字小文字変換が実現されます。UCA はユニコード規格の一部として開発されました。SQL Anywhere では、IBM が開発および保守している International Components for Unicode (ICU) オープン・ソース・ライブラリを使用して UCA を実装しています。

Windows CE では、NCHAR 照合または CHAR 照合として UCA 使用している場合は、ICU が必要です。これは、UCA が ICU を必要とするからです。また、CHAR 文字セットがオペレーティング・システムの文字セットと一致しない場合も、Windows CE で ICU が必要です。

デフォルトでは、ICU ライブラリは Windows CE にインストールされません。これは、このライブラリをインストールすると、Windows CE に SQL Anywhere をインストールするときのサイズが約 1.7 MB 大きくなるからです。ただし、これらのライブラリが必要になったら、後で SQL Anywhere インストールを変更できます。

ICU ライブラリをインストールしていない場合は、Windows CE の文字セットと一致する文字セットを使う照合を選択するか、データベースを作成するときに CHAR 照合として UTF8BIN 照合を選択することが必要です。また、データベースを作成するときに NCHAR 照合として UTF8BIN を選択することも必要になります。

デスクトップにデータベースを作成して Windows CE に配備
デスクトップにデータベースを作成して Windows CE に配備するときは、UCA 照合のみを使用するようにします (Windows CE デバイスに ICU ライブラリがインストールされている場合)。ICU ライブラリがインストールされていない場合、UCA を使うデータベースを Windows CE で利用することはできません。

ICU の詳細については、「ユニコード照合アルゴリズム (UCA)」 350 ページと「文字セット変換」 344 ページを参照してください。

インストール時の考慮事項： Windows CE での .NET Compact Framework の使用

API の最新バージョンは ADO.NET 2.0 ですが、SQL Anywhere がサポートする大部分のデバイスには ADO.NET 1.x サポートのみがインストールされています。デバイス上で ADO.NET バージョン 2.0 を使用するには、Microsoft 社の Web サイトから ADO.NET 2.0 サポートをダウンロードし、デバイスにインストールする必要があります。

ADO.NET アプリケーションを開発する予定がある場合は、SQL Anywhere をインストールするときに、使用する .NET Compact Framework のバージョンを指定することが必要です。

- **バージョン 1.x** これはデフォルトです。ADO.NET 1.0 を使用してアプリケーションを開発する場合の詳細については、http://www.ianywhere.com/downloads/products/sqlanywhere/sql_10_dotnet_api_reference.pdf の SQL Anywhere .NET データ・プロバイダの API リファレンスを参照してください。
- **バージョン 2.0** ADO.NET 2.0 を使用したアプリケーション開発の詳細については、「SQL Anywhere .NET データ・プロバイダ」 『SQL Anywhere サーバ - プログラミング』と「SQL Anywhere .NET 2.0 API リファレンス」 『SQL Anywhere サーバ - プログラミング』を参照してください。

ADO.NET の使用の詳細については、「チュートリアル：SQL Anywhere .NET データ・プロバイダの使用」 『SQL Anywhere サーバ - プログラミング』を参照してください。

インストール時の考慮事項： Windows Mobile 5 への SQL Anywhere 10 の配備

Windows Mobile 5 はデジタル・コードでの署名を使用して、デジタル・コンテンツの出所を確認します。SQL Anywhere を Windows Mobile 5 デバイスに配備しているときは、以下のいずれかのタイプの *.cab* ファイルを配備できます。

- **構築済みの、署名された .CAB ファイル**　SQL Anywhere インストールには、VeriSign によって署名された、構築済みの *.cab* ファイルが含まれます。これらの *.cab* ファイルを使用しても、発行元が不明であることを知らせる警告は表示されません。これらのファイルは、.NET Compact Framework のバージョン 1.1 と 2.0 の両方で提供されます。
- **カスタムの、署名されていない .CAB ファイル**　カスタムの、署名されていない *.cab* ファイルを構築し、サポートされている Windows CE プラットフォームに SQL Anywhere を配備できます。Windows Mobile 5 で、カスタムの、署名されていない *.cab* ファイルを実行すると、発行元が不明であることを知らせる警告が表示されますが、ソフトウェアのインストールには影響しません。

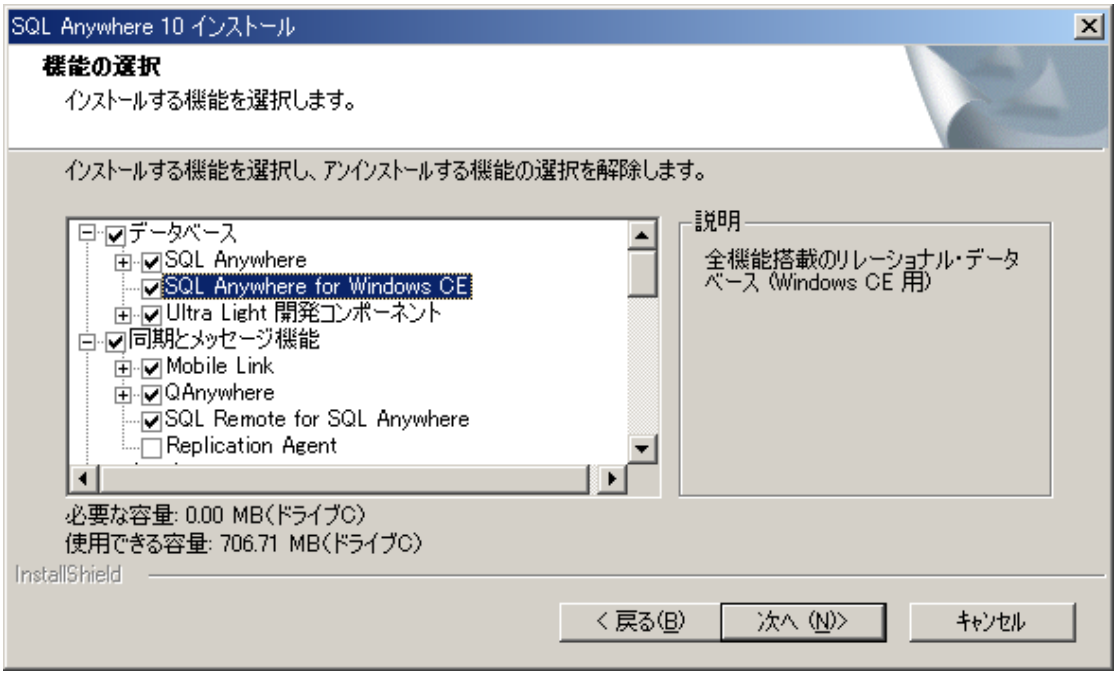
SQL Anywhere for Windows CE のインストール

ここでは、Windows CE デバイスに SQL Anywhere for Windows CE をインストールする手順について説明します。

- **SQL Anywhere for Windows CE をインストールするには、次の手順に従います。**

1. サポートされている Windows オペレーティング・システムを実行しているコンピュータに、Windows CE デバイスを接続します。
2. コンピュータで SQL Anywhere *setup.exe* を実行して、インストールを開始します。
3. [SQL Anywhere 10 インストール] ウィザードの指示に従います。
4. [コンポーネントの選択] ページで、[データベース] の下にある [SQL Anywhere for Windows CE] チェックボックスをオンにします。

   このバージョンの SQL Anywhere をすでにコンピュータにインストールしている場合は、[データベース] の下にある [SQL Anywhere] チェックボックスをオフにして、ソフトウェアがコンピュータに再インストールされないようにします。



5. ウィザードの残りの指示に従います。

Windows CE サンプル・アプリケーションの使用

この項では、SQL Anywhere for Windows CE に付属のサンプル・データベースとサンプル・アプリケーションについて説明します。

サンプル・データベースは、限られた種類のスポーツ衣料品を製造する小企業を想定して作られています。

サンプル・データベースは、「*demo.db*」というファイルに入っています。Windows CE デバイスでは、このファイルは *¥Program Files¥SQLAny10* ディレクトリにあります。Windows CE では、2 つのバージョンのサンプル・データベースを利用できます。1 つのバージョンには、ICU (International Components for Unicode) ライブラリが含まれています。もう 1 つのバージョンには含まれていません。SQL Anywhere では、IBM が開発および保守している ICU オープン・ソース・ライブラリを使用して文字セット変換を実装しています。

ICU および Windows CE の詳細については、「インストール時の考慮事項： Windows CE での ICU の使用」 1028 ページを参照してください。

SQL Anywhere for Windows CE のインストールには、次のサンプル・アプリケーションが含まれています。

- ADOCE Sample
- ADO.NET Sample
- ESQL Sample
- ODBC Sample
- SQL Anywhere Server Example

これらのアプリケーションを使用すると、サンプル・データベースにアクセスして、SQL Anywhere for Windows CE の機能を検証できます。

SQL Anywhere Server Example

SQL Anywhere Server Example は、事前に設定されたサーバ・オプションと接続パラメータを使用して、ネットワーク・データベース・サーバとしてサンプル・データベースを起動します。

♦ SQL Anywhere Server Example を起動するには、次の手順に従います。

1. Windows CE デバイスで、[スタート] メニューから、[プログラム] – [SQLAny10] の順にタップして、SQL Anywhere インストール・ディレクトリに移動します。
2. SQL Anywhere Server Example をタップして、ネットワーク・データベース・サーバ上のサンプル・データベースを起動します。

   ネットワーク・データベース・サーバでサンプル・データベースが開始します。開始すると、データベース・サーバは Windows CE デバイスの [Today] 画面の右下にアイコンとして表示されます。このアイコンをタップすると、サーバ・メッセージ・ウィンドウを表示できます。

これで、コンピュータから Windows CE デバイスのサンプル・データベースに接続できます。

サンプル・データベースを使い終わったら、データベース・サーバを停止してください。

♦ **データベース・サーバを停止するには、次の手順に従います。**

1. [Today] 画面の右下にあるネットワーク・データベース・サーバ・アイコンをタップします。

   サーバ・メッセージ・ウィンドウが表示されます。

2. [シャットダウン] をタップします。

ADO.NET Sample

ADO.NET Sample を使用するには、デバイスに Microsoft .NET Compact Framework バージョン 1.0 と Service Pack 2 以降をインストールしておく必要があります。

このコンポーネントは、http://www.microsoft.com/downloads/search.aspx?displaylang=ja からダウンロードできます。

ADO.NET Sample は、ADO.NET プログラミング・インタフェースを使用する簡単なアプリケーションのデモです。このアプリケーションでは、ネットワーク・データベース・サーバで実行中のサンプル・データベースを起動し、SQL 文でアクセスして、データを変更できます。

このサンプルのソース・コードは、*samples-dir¥SQLAnywhere¥ce¥ado_net_sample* にあります。

Visual Studio では、*samples-dir¥SQLAnywhere¥ce¥ado_net_sample¥ado_net_sample.sln* からこのプロジェクトをロードできます。

注意
ADO.NET Sample のユーザ・インタフェースでは、SQL 文を 1 行で入力する必要があります。

♦ ADO.NET Sample を使用するには、次の手順に従います。

1. [スタート] メニューから、[プログラム] − [SQLAny10] − [ADO.NET Sample] の順にタップして、ADO.NET Sample を開始します。
2. [Connect] をタップします。

   ネットワーク・データベース・サーバでサンプル・データベースが開始します。サーバ・ウィンドウが表示されて、すぐに閉じます。
3. [Exec SQL] をタップして、デフォルトの SQL 文「SELECT * FROM Employees」を実行してから、[Exec SQL] をタップします。

   Employees テーブルのすべてのデータが、データ・ウィンドウに表示されます。
4. データ・ウィンドウの右と下にあるスクロール・バーを使って、Employees テーブルのデータ間を移動します。
5. 次のような、特定のデータ範囲にアクセスするクエリを入力します。

   ```
   SELECT EmployeeID, Surname FROM Employees
   ```
6. [Exec SQL] をタップして、SQL 文を実行します。

   指定したデータ範囲が、データ枠の元のデータを置き換えます。
7. 「SELECT * FROM Employees ORDER BY EmployeeID」と入力し、[Exec SQL] をタップします。

   EmployeeID が 105 の従業員 Matthew Cobb に注目します。
8. 「UPDATE Employees SET Surname = 'Jones' WHERE Surname = 'Cobb'」と入力し、[Exec SQL] をタップして SQL 文を実行します。
9. 「SELECT * FROM Employees ORDER BY EmployeeID」と入力し、[Exec SQL] をタップします。

   Matthew の姓が Cobb から Jones に変わりました。
10. 「UPDATE Employees SET Surname = 'Cobb' WHERE Surname = 'Jones'」と入力し、[Exec SQL] をタップして、サンプル・データベースに加えた変更を元に戻します。
11. 「SELECT * FROM Employees ORDER BY EmployeeID」と入力し、[Exec SQL] をタップして、内容が元に戻ったことを確認します。

    Matthew の姓が Jones から元の Cobb に戻りました。

12. [Exec SQL] フィールドに「SELECT * FROM Customers」と入力し、[Exec SQL] をタップして、別のテーブルのデータにアクセスします。

    Customers テーブルのすべてのデータがデータ・ウィンドウに表示されて、Employees テーブルのデータを置き換えます。

13. [Disconnect] をタップして、データベース・サーバを停止します。

    ADO.NET Sample が切断され、データベース・サーバが自動的に停止します。

14. ウィンドウの右上隅の [X] をタップして、ADO.NET Sample を閉じます。

ADOCE Sample

ADOCE Sample は、Windows CE 開発用 ADO プログラミング・インタフェース、ADOCE を使用する簡単なアプリケーションのデモです。このアプリケーションでは、ネットワーク・データベース・サーバで実行中のサンプル・データベースを起動し、SQL 文を使用してデータにアクセスできます。

ADOCE sample を使用するには、Windows CE デバイスに Visual Basic をインストールしておくことが必要です。

このサンプルのソース・コードは、コンピュータの *samples-dir¥SQLAnywhere¥ce¥adoce_sample* にあります。

eMbedded Visual Basic では、コンピュータの *samples-dir¥SQLAnywhere¥ce¥adoce_sample* からこのプロジェクトをロードできます。

注意
ADOCE Sample のユーザ・インタフェースでは、SQL 文を 1 行で入力する必要があります。

♦ ADOCE Sample を使用するには、次の手順に従います。

1. [スタート] メニューから、[プログラム] – [SQLAny10] の順にタップして、ADOCE Sample を開始します。
2. [ADOCE Sample] をタップします。
3. [Connect] をタップします。

   ネットワーク・データベース・サーバでサンプル・データベースが開始します。サーバ・ウィンドウが表示されて、すぐに閉じます。

4. [Exec SQL] をタップして、デフォルトの SQL 文「SELECT * FROM Employees」を実行します。

   Employees テーブルのすべてのデータが、データ・ウィンドウに表示されます。

5. データ・ウィンドウの端にあるスクロール・バーを使って、Employees テーブルのデータ間を移動します。

6. 「SELECT EmployeeID, Surname FROM Employees」」と入力し、[Exec SQL] をタップして文を実行して、特定のデータ範囲にアクセスします。
7. 下方にスクロールしてデータを見ます。データは、データ・ウィンドウにすでに表示されているデータの下にあります。
8. [Exec SQL] フィールドに「SELECT * FROM Customers」と入力し、[Exec SQL] をタップして、別のテーブルのデータにアクセスします。
9. Customers テーブルから下方にスクロールしてデータを見ます。データは、データ・ウィンドウにすでに表示されているデータの下にあります。
10. [Disconnect] をタップして、ネットワーク・データベース・サーバを停止します。

    ADOCE Sample が切断され、ネットワーク・データベース・サーバが自動的に停止します。
11. ウィンドウの右上隅の [X] をタップして、ADOCE Sample を閉じます。

ESQL Sample

ESQL Sample は、Embedded SQL プログラミング・インタフェースを使用する簡単なアプリケーションのデモです。このアプリケーションでは、ネットワーク・データベース・サーバで実行中のサンプル・データベースを起動し、SQL 文を使用してデータにアクセスできます。

このサンプルのソース・コードは、*samples-dir¥SQLAnywhere¥ce¥esql_sample* にあります。

Visual Studio では、*samples-dir¥SQLAnywhere¥ce¥sql_sample¥esql_sample.sln* からこのプロジェクトをロードできます。

注意
ESQL Sample のユーザ・インタフェースでは、SQL 文を 1 行で入力する必要があります。

♦ ESQL Sample を使用するには、次の手順に従います。

1. [スタート] メニューから、[プログラム] – [SQLAny10] の順にタップして、ESQL Sample を開始します。
2. [ESQL Sample] をタップします。
3. [Connect] をタップし、デフォルトの接続文字列を使ってサンプル・データベースに接続します。

   ネットワーク・データベース・サーバでサンプル・データベースが開始します。サーバ・ウィンドウが表示されて、すぐに閉じます。
4. [Exec SQL] をタップして、デフォルトの SQL 文「SELECT * FROM Employees」を実行します。

   Employees テーブルのすべてのデータが、データ・ウィンドウに表示されます。

5. データ・ウィンドウの右と下にあるスクロール・バーを使って、Employees テーブルのデータ間を移動します。
6. [Exec SQL] フィールドに「SELECT * FROM Customers」と入力し、[Exec SQL] をタップして、別のテーブルのデータにアクセスします。

   Customers テーブルのすべてのデータがデータ・ウィンドウに表示されて、Employees テーブルのデータを置き換えます。
7. [Disconnect] をタップして、ネットワーク・データベース・サーバを停止します。

   ESQL Sample が切断され、ネットワーク・データベース・サーバが自動的に停止します。
8. ウィンドウの右上隅の [X] をタップして、ESQL Sample を閉じます。

ODBC Sample

ODBC Sample は、ODBC プログラミング・インタフェースを使用する簡単なアプリケーションのデモです。このアプリケーションでは、ネットワーク・データベース・サーバで実行中のサンプル・データベースを起動し、基本的な SQL 文を使用してデータにアクセスできます。

このサンプルのソース・コードは、*samples-dir¥SQLAnywhere¥ce¥odbc_sample* にあります。

Visual Studio 2005 では、*samples-dir¥SQLAnywhere¥ce¥odbc_sample¥odbc_sample.sln* からこのプロジェクトをロードできます。

注意
ODBC Sample のユーザ・インタフェースでは、SQL 文を 1 行で入力する必要があります。

♦ **ODBC Sample を使用するには、次の手順に従います。**

1. [スタート] メニューから、[プログラム] − [SQLAny10] − [ODBC Sample] の順にタップして、ODBC Sample を開始します。
2. [Connect] をタップします。

   ネットワーク・データベース・サーバでサンプル・データベースが開始します。サーバ・ウィンドウが表示されて、すぐに閉じます。
3. [Exec SQL] をタップして、デフォルトの SQL 文「SELECT * FROM Employees」を実行します。

   Employees テーブルのすべてのデータが、データ・ウィンドウに表示されます。
4. データ・ウィンドウの右と下にあるスクロール・バーを使って、Employees テーブルのデータ間を移動します。
5. [Exec SQL] フィールドに「SELECT * FROM Customers」と入力し、[Exec SQL] をタップして、別のテーブルのデータにアクセスします。

Customers テーブルのすべてのデータがデータ・ウィンドウに表示されて、Employees テーブルのデータを置き換えます。

6. [Disconnect] をタップして、ネットワーク・データベース・サーバを停止します。

   ODBC Sample が切断され、ネットワーク・データベース・サーバが自動的に停止します。

7. ウィンドウの右上隅の [X] をタップして、ODBC Sample を閉じます。

Windows CE デバイスで実行中のデータベースへの接続

コンピュータ上で実行しているアプリケーションを、Windows CE デバイス上で実行しているデータベースに接続するには、コンピュータと Windows CE デバイス間の ActiveSync リンクを介した TCP/IP 接続によって実現できます。これにより、コンピュータ上の管理ユーティリティを使用して、Windows CE データベースを管理できます。

参照

- 「Windows CE での管理ユーティリティの使用」 1058 ページ

Windows CE デバイス上でのサーバの起動

デスクトップ・コンピュータから Windows CE で実行されているデータベース・サーバに接続するには、サーバを起動するときに TCP/IP オプションを選択する必要があります。

USB 経由の ActiveSync 接続が可能である場合は (デバイスのクレードルが接続されている場合など)、Windows CE データベース・サーバに接続するには、プロキシ・ポートを設定する必要があります。

♦ リモート接続用に、Windows CE でデータベース・サーバを起動するには、次の手順に従います。

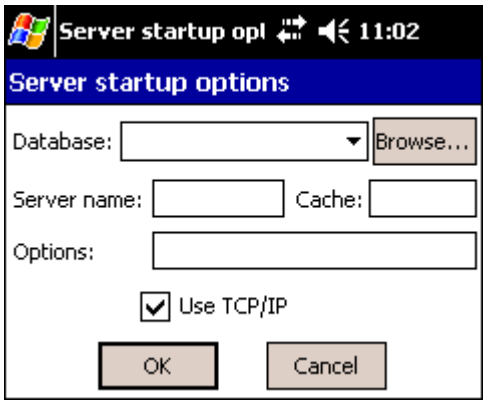
1. Windows CE デバイスで、[ファイル エクスプローラ] を開きます。

   [スタート] メニューから、[プログラム] – [ファイル エクスプローラ] の順に選択します。

2. [プログラム] – [SQLAny10] の順にタップして、SQL Anywhere インストール・ディレクトリに移動します。

3. *dbsrv10* アイコンをタップします。

   [サーバ起動オプション] ダイアログが表示されます。



4. 開始するデータベース・ファイルを指定します。

[参照] ボタンを使用して、データベースを検索します。

5. 使用するサーバ名を指定します。

   たとえば、[サーバ起動オプション] ダイアログの [サーバ名] フィールドに **CEserver** と入力します。

6. [TCP/IP を使用] を選択します。

   コンピュータから Windows CE デバイスで実行中のデータベースに接続するには、TCP/IP 接続が必要です。

7. デフォルトである 2638 以外のプロキシ・ポート番号を使用するには、**-x tcpip** オプションを使用してプロキシ・ポート番号を指定します。たとえば、プロキシ・ポート番号が 2639 である場合、[サーバ起動オプション] ダイアログの [オプション] フィールドに **-x tcpip {port=2639}** と入力します。

   -x tcpip オプションを使用する場合に [TCP/IP を使用] オプションを選択すると、設定が重複することに注意してください。

   詳細については、「-x サーバ・オプション」 205 ページを参照してください。

8. [OK] をタップして、ネットワーク・データベース・サーバで実行中のサンプル・データベースを開始します。

これで、プロキシ・ポートと ODBC データ・ソースを作成して、コンピュータから Windows CE デバイスに接続できます。

参照

- 「Windows CE デバイス用プロキシ・ポートの作成」 1040 ページ
- 「Windows CE デバイスに接続するための ODBC データ・ソースの作成」 1041 ページ

Windows CE デバイス用プロキシ・ポートの作成

Windows CE デバイスでデータベースを実行するには、Windows CE デバイスで実行中のデータベースに要求が渡されるように、ActiveSync 用のプロキシ・ポートを設定する必要があります。

♦ Windows CE デバイス用プロキシ・ポートを作成するには、次の手順に従います (Interactive SQL の場合)。

1. Interactive SQL を開きます。
2. [SQL] メニューから [接続] を選択します。

   [接続] ダイアログが表示されます。

3. [ツール] をクリックし、[Windows CE プロキシ・ポートの設定] を選択します。

   [Windows CE プロキシ・ポート] ダイアログが開きます。

4. [新規] をクリックします。

[新しいプロキシ・ポート] ダイアログが表示されます。

5. [名前] フィールドに「**SQL Anywhere**」と入力します。
6. [ポート] フィールドに「**2638**」と入力します。

   これは SQL Anywhere のデフォルトの TCP/IP ポートです。

> **注意**
> Windows CE デバイスをクレードルに置くたびに、ActiveSync がポート 2638 のトラフィックをデバイスに転送します。Windows CE デバイスがクレードルにある間にコンピュータでデータベース・サーバを起動すると、デフォルト・ポートの 2638 を使用できません。これが問題になる場合は、Windows CE トラフィック専用に別のポートを選択することもできます。

7. プロキシ・ポートの設定や変更を行うとき、プロキシ・ポートの新しい設定を有効にするために Windows デスクトップをログオフして、もう一度ログオンすることが必要になる場合があります。

レジストリ・キー H*KEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows CE Services ¥ProxyPorts* を設定して、プロキシ・ポートを設定することもできます。.DWORD 値は **SQL Anywhere** とし、Windows CE トラフィック専用のポートを指定します (2638 を選択することをおすすめします)。

> **警告**
> レジストリの変更は危険です。レジストリは、ユーザ自身の責任で変更してください。

Windows CE デバイスに接続するための ODBC データ・ソースの作成

この項では、Windows コンピュータで ODBC データ・ソースを作成して、Windows CE デバイスで実行中のデータベースに接続する方法を説明します。

ODBC データ ソースの詳細については、「ODBC データ・ソースの使用」 75 ページを参照してください。

♦ ODBC データ・ソースを作成して Windows CE デバイスに接続するには、次の手順に従います。

1. コンピュータで、ODBC アドミニストレータを開きます。

   [スタート] – [プログラム] – [SQL Anywhere 10] – [SQL Anywhere] – [ODBC アドミニストレータ] を選択します。

   [ODBC アドミニストレータ] が表示されます。

2. [ユーザー DSN] タブで、[追加] をクリックします。

   [データ ソースの新規作成] ダイアログが表示されます。

3. [SQL Anywhere 10] を選択して、[完了] をクリックします。

   [ODBC 設定] ダイアログが表示されます。

4. [ODBC] タブの [データ・ソース名] フィールドに、データ・ソースの名前を入力します。

   たとえば、「**CEdevice**」と入力します。

5. [ログイン] タブで、[ユーザ ID とパスワードを指定] を選択します。[ユーザ ID] と [パスワード] フィールドは空白のままにします。

   データベースに接続するたびに、ユーザ ID とパスワードを指定する必要があります。

6. [データベース] タブで、[サーバ名] フィールドを空白のままにします。

   コンピュータから接続するたびに、サーバ名を指定する必要があります。この名前は、Windows CE デバイスのサーバ・メッセージ・ウィンドウのタイトル・バーに表示されます。

7. [ネットワーク] タブで、[TCP/IP] チェックボックスをオンにします。

8. 隣のフィールドに接続パラメータを入力します。

   たとえば、「**Host=127.0.0.1;Port=2638;DoBroadcast=none**」と入力します。

   - **Host** Windows CE デバイスが受信する IP アドレスを指定します。

     Windows CE デバイスに接続するためのプロキシ・ポートを作成している場合は、デフォルトの IP アドレス **127.0.0.1** (デフォルトのローカル・ホスト・アドレス) を使用します。

     詳細については、「Windows CE デバイス用プロキシ・ポートの作成」 1040 ページを参照してください。

     ポートを作成していない場合は、Windows CE デバイスの IP アドレスを使用します。

     詳細については、「Windows CE デバイス上でのサーバの起動」 1039 ページを参照してください。

   - **Port** Windows CE デバイスが受信するポート番号を指定します。このパラメータはオプションです。デフォルトのプロキシ・ポート番号を使用しない場合には、指定する必要があります。

     プロキシ・ポート番号が指定されない場合、**2638** が指定されたと見なされます。

     詳細については、「Windows CE デバイス用プロキシ・ポートの作成」 1040 ページを参照してください。

   - **DoBroadcast** TCP/IP 接続の確立方法を制御します。このパラメータはオプションです。

     「**DoBroadcast=none**」と指定すると、指定されているポートと直接 TCP/IP 接続が確立されます。Windows CE デバイスに接続するためのプロキシ・ポートを作成している場合は、この設定を使用します。

詳細については、「Windows CE デバイス用プロキシ・ポートの作成」 1040 ページを参照してください。

「**DoBroadcast=DIRECT**」と指定した場合は、データベース・サーバを検索するときにローカル・サブネットへのブロードキャストは実行されません。ホストの IP アドレスが必要になります。

詳細については、「DoBroadcast プロトコル・オプション [DOBROAD]」 271 ページを参照してください。

9. [OK] をクリックして、データ・ソースを作成します。

これで、このデータ・ソースを使用して、コンピュータから、Windows CE 上で実行中のデータベースに接続できます。

詳細については、「Windows CE での管理ユーティリティの使用」 1058 ページを参照してください。

Windows CE デバイスの IP アドレスの特定

Windows CE で実行中のデータベースへの接続を確立するには、IP アドレスが必要になることがあります。

USB 経由の ActiveSync 接続が可能である場合は (デバイスのクレードルが接続されている場合など)、Windows CE データベースに接続するには、プロキシ・ポートを設定する必要があります。「Windows CE デバイス用プロキシ・ポートの作成」 1040 ページを参照してください。

♦ Windows CE デバイスの IP アドレスを特定するには、次の手順に従います。

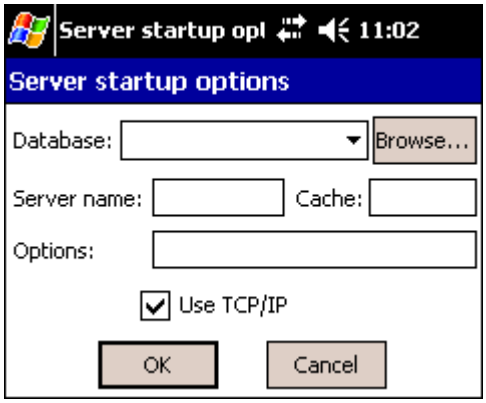
1. Windows CE デバイスで、[ファイル エクスプローラ] を開きます。

   [スタート] メニューから、[プログラム] – [ファイル エクスプローラ] の順に選択します。

2. [プログラム] – [SQLAny10] の順にタップして、SQL Anywhere インストール・ディレクトリに移動します。

3. *dbsrv10* アイコンをタップします。

   [サーバ起動オプション] ダイアログが表示されます。



4. 開始するデータベース・ファイルを指定します。

   [参照] ボタンを使用して、データベースを検索します。

5. 使用するサーバ名を指定します。

   たとえば、[サーバ起動オプション] ダイアログの [サーバ名] フィールドに **CEserver** と入力します。

6. [TCP/IP を使用] を選択します。

   コンピュータから Windows CE デバイスで実行中のデータベースに接続するには、TCP/IP 接続が必要です。

7. [サーバ起動オプション] ダイアログの [オプション] フィールドに、「**-z**」と入力します。

   サーバは -z オプションを使用して、起動中に IP アドレスを書き出します。Windows CE デバイスをネットワークから切断して再接続すると、アドレスが変わることがあります。

   詳細については、「-z サーバ・オプション」 210 ページを参照してください。

8. [OK] をタップして、ネットワーク・データベース・サーバで実行中のサンプル・データベースを開始します。

9. デバイスの [Today] 画面に移動します。

10. 画面の右下にあるサーバ・アイコンをタップします。

    サーバ・メッセージ・ウィンドウが表示されます。IP アドレスはサーバ・メッセージ・ウィンドウに表示されます。

これで、ODBC データ・ソースを作成して、コンピュータから Windows CE デバイスに接続できます。

詳細については、「Windows CE デバイスに接続するための ODBC データ・ソースの作成」 1041 ページを参照してください。

Windows CE データベースの設定

SQL Anywhere の完全バージョンで使用できる SQL 機能のほとんどは、Windows CE バージョンでも使用できます。たとえば、トランザクション処理、参照整合性アクション、プロシージャ、トリガなどです。ただし、Java の機能とリモート・データ・アクセス機能は、Windows CE では使用できません。

Windows CE デバイスで使用するデータベースのプロパティを設定するときは、サポートされていない機能に注意してください。

機能の完全なリストについては、「Windows CE での SQL Anywhere 機能のサポート」 1066 ページを参照してください。

以下の設定は、データベースの作成中に設定します。いったん設定すると、データベースを再構築しなければ変更できません。

- **大文字と小文字の区別**　「大文字と小文字の区別」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **比較時の後続ブランクの処理**　デフォルトでは、後続ブランクを余分の文字と見なしてデータベースが作成されます。たとえば、'Dirk' は 'Dirk ' と同じではありません。後続ブランクが無視されるように、ブランクを埋め込んだデータベースを作成できます。「比較の後続ブランクを無視」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **ページ・サイズ**　「テーブルとページのサイズ」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **照合順と文字セット**　Windows CE 用のデータベースを作成するときは、Windows が対象の言語に使用するのと同じシングルバイト文字セットまたはマルチバイト文字セットを基にした照合を使用してください。たとえば、英語、フランス語、またはドイツ語を使用する場合は、1252Latin1 照合を使用します。日本語を使用する場合は 932JPN 照合を、韓国語を使用する場合は 949KOR 照合を使用します。「照合の知識」 349 ページを参照してください。

> **注意**
> Windows CE で使用するデータベースを作成するときは、適合化オプションの文字列にロケールまたはソート・タイプを指定しないでください。指定すると、Windows CE デバイスでデータベースが起動しなくなる可能性があります。照合の適合化オプションの詳細については、「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

文字セット変換は Windows CE ではサポートされていないため、Windows CE 上のデータベースには、オペレーティング・システムの文字セットと UTF-8 のいずれかを使用する必要があります。

Windows CE データベースを作成するときに、ICU ライブラリをインストールするかどうかを選択することが必要です。「インストール時の考慮事項： Windows CE での ICU の使用」 1028 ページを参照してください。

Windows CE でのトランザクション・ログの使用

トランザクション・ログには、データベースに加えられた変更がその変更順に格納されます。データベース・ファイルでメディア障害が発生した場合、トランザクション・ログはデータベースのリカバリに重要な役割を果たします。また、トランザクション・ログを使用すると、より効率的に作業できます。トランザクション・ログは、デフォルトではデータベース・ファイルと同じディレクトリに配置されます。このログは、Windows CE デバイスで初めてデータベースを開始するときに作成されます。

既存のデータベースを Windows CE デバイスにコピーする場合は、データベース・ファイルとトランザクション・ログ・ファイルの両方をコピーできます。トランザクション・ログ・ファイルをデバイスにコピーしなければ、Windows CE デバイスでデータベースを開始したときに新しいログが作成されます。新しいトランザクション・ログには、元のトランザクション・ログに格納されていた情報は含まれません。これは、最後にデータベースを使用したときに正しく停止しなかった場合や、データベースを同期している場合に問題になることがあります。データベース・ファイルとトランザクション・ログ・ファイルの両方を Windows CE デバイスにコピーすることをおすすめします。

参照

- 「トランザクション・ログ」 816 ページ

Windows CE での jConnect の使用

jConnect は、SQL Anywhere 用の純正 Java JDBC ドライバです。Sybase Central と Interactive SQL には、Java アプリケーションで SQL Anywhere のデータベースにアクセスできるように、jConnect JDBC ドライバを有効にするオプションがあります。

Windows CE 用に作成されたデータベースでは、デフォルトでは jConnect は有効になっていません。ただし、必要に応じて、jConnect を有効にすることができます。

データベースに jConnect のサポートを追加すると、多数のエントリがシステム・テーブルに追加されます。これによってデータベース・サイズが大きくなり、たとえ jConnect 機能を使用しない場合でも、データベースの実行に必要なメモリが約 200 KB 増えます。

Windows CE のような限られたメモリ環境で作業をする場合、jConnect を使用する必要がなければ、データベースに jConnect のサポートを追加することはおすすめしません。

参照

- 「jConnect JDBC ドライバの使用」 『SQL Anywhere サーバ - プログラミング』

Windows CE での暗号化の使用

データベースを安全に管理するために、単純暗号化または強力な暗号化のいずれかを選択できます。データベースを初期化した後で暗号化の設定を変更するには、データベース全体を再構築するしかありません。

参照

- 「データベースの暗号化」 936 ページ
- 「Windows CE データベースの保護」 947 ページ

Windows CE データベースの作成

Windows CE デバイス用に SQL Anywhere データベースを作成する方法は、3 通りあります。

- Sybase Central の [データベース作成] ウィザードを使用して、Windows CE デバイスに直接コピーできるデータベースを作成する
- 初期化ユーティリティ (dbinit) を使用して、Windows CE デバイスに手動でコピーできるデータベースを作成する
- Interactive SQL で CREATE DATABASE 文を使用して、Windows CE デバイスに手動でコピーできるデータベースを作成する

Windows CE データベース用にチェックサムを有効化
Windows CE 上に配備するデータベースを作成する場合は、チェックサムを有効にする必要があります。これによって、データベース・ファイルの破損を速やかに検出できます。

Windows CE データベースを作成するときに決定することが必要な事項については、次の項を参照してください。

- 「Windows CE でのトランザクション・ログの使用」 1046 ページ
- 「インストール時の考慮事項： Windows CE での ICU の使用」 1028 ページ
- 「インストール時の考慮事項： Windows CE での .NET Compact Framework の使用」 1029 ページ

Sybase Central を使用した Windows CE データベースの作成

Sybase Central には、Windows CE データベース用に簡単にデータベースを作成する機能があります。Windows コンピュータに Windows CE サービスをインストールしている場合は、Sybase Central を使用して Windows CE データベースを作成できます。Sybase Central は、Windows CE のデータベースに必要な条件を満たし、生成されたデータベース・ファイルを Windows CE デバイスにコピーするオプションを提供します。

Windows CE データベース用にチェックサムを有効化
Windows CE 上に配備するデータベースを作成する場合は、チェックサムを有効にする必要があります。これによって、データベース・ファイルの破損を速やかに検出できます。

♦ Sybase Central で Windows CE のデータベースを作成して、それを Windows CE デバイスに直接コピーするには、次の手順に従います。

1. Windows CE デバイスとコンピュータが接続されていることを確認します。
2. [スタート] メニューから、[プログラム] – [SQL Anywhere 10] – [Sybase Central] を選択して、コンピュータ上で Sybase Central を起動します。
3. [ツール] – [SQL Anywhere 10] – [データベースの作成] を選択し、Sybase Central の [データベース作成] ウィザードを起動します。

   [データベース作成] ウィザードが表示されます。
4. [このコンピュータにデータベースを作成] を選択し、[次へ] をクリックします。

   データベースは、Windows CE デバイスにコピーされる前にコンピュータで初期化されます。
5. データベース・ファイルを格納するコンピュータのディレクトリとファイル名を指定し、[次へ] をクリックします。
6. [Windows CE にこのデータベースを作成] を選択し、[次へ] をクリックします。

   このオプションを使用すると、新しいデータベースが Windows CE に対応した設定になります。

   [次へ] をクリックすると、Windows CE デバイスへの接続が開始されます。
7. [データベースを Windows CE デバイスにコピー] を選択すると、デバイスが初期化されたあと、自動的にデータベースがデバイスにコピーされます。[次へ] をクリックします。
8. データベース・ファイルをコピーする Windows CE のディレクトリを指定します。デフォルト・ロケーションは、メイン・デバイスのディレクトリです。

   ヒント
   Windows CE デバイスの My Documents ディレクトリにデータベースをコピーすると、データベースの開始が容易になります。
   Windows CE デバイスで [サーバ起動オプション] ダイアログを使用してデータベースを開始するときに、参照ボタンを使用すると My Documents ディレクトリ内のデータベース・ファイルの検索しかできません。
   データベースを My Documents ディレクトリに保存していない場合は、[サーバ起動オプション] ダイアログの [データベース] フィールドにデータベースのパスを入力する必要があります。

   オプションとして、[コピー後にデスクトップのデータベースを削除] を選択できます。

   コンピュータのコピーを削除しない場合は、手順 5 で指定したディレクトリにデータベース・ファイルのコピーが格納されます。[次へ] をクリックします。

9. [NCHAR データの照合順の指定] ページで、データベースに使用する NCHAR 文字セットの照合を選択します。文字セット変換を使用しておらず、ICU ライブラリが必要ない場合は、UTF8BIN を選択します。

10. トランザクション・ログ・ファイルを保存するディレクトリを指定します。[次へ] をクリックします。

    Windows CE デバイスでは、ネットワーク・データベース・サーバで初めてデータベースを開始したときに、データベース・ファイルと同じディレクトリにトランザクション・ログ・ファイルが作成されます。

11. トランザクション・ログ・ミラーを使用するかどうかを指定します。[次へ] をクリックします。

    ミラー・ログは、万一トランザクション・ログが破損した場合のデータ・リカバリに役立ちますが、必須ではありません。

12. [jConnect メタ情報サポートをインストール] オプションが選択されていないことを確認します。[次へ] をクリックします。

13. 適切なオプションを選択して、データベースの暗号化レベルを設定します。[次へ] をクリックします。

    強力な暗号化を選択した場合は、暗号化キーを指定する必要があります。キーには最低でも 16 文字の値を選択し、大文字と小文字、数字、文字、特殊文字を組み合わせて使用することをおすすめします。

    警告
    キーのコピーは安全な場所に保管してください。キーは、データベースを起動したり変更したりするたびに必要になります。キーを紛失すると、データベースにまったくアクセスできなくなり、リカバリも不可能になります。

14. [各データベース・ページでチェックサムを含む] オプションが選択されていることを確認します。このページでは、必要に応じて、他のオプションを選択できます。[次へ] をクリックします。

    ウィザードの残りの指示に従います。

15. [完了] をクリックすると、データベースが作成されてデバイスにコピーされます。

    Windows CE デバイスにコピーしているファイルの進行状況を示すダイアログが表示されます。

16. データベースが Windows CE デバイスにコピーされたら、ファイルの場所を確認します。

    [スタート] メニューから [プログラム] – [ファイル エクスプローラ] の順にタップして、データベースをコピーした Windows CE ディレクトリに移動します。

    データベース・ファイルが表示されます。トランザクション・ログ・ファイルは、Windows CE デバイスで初めてデータベースを開始するときまで表示されません。

dbinit を使用した Windows CE データベースの作成

Windows CE で使用できるデータベースを作成するために、初期化ユーティリティ (dbinit) を使用できます。ただし、このユーティリティから Windows CE デバイスに直接データベースをコピーすることはできません。dbinit ユーティリティを使用して作成したデータベースは、Windows CE デバイスに手動でコピーすることが必要です。

Windows CE データベース用にチェックサムを有効化
Windows CE 上に配備するデータベースを作成する場合は、チェックサムを有効にする必要があります。これによって、データベース・ファイルの破損を速やかに検出できます。

♦ **dbinit ユーティリティを使用してデータベースを作成するには、次の手順に従います。**

1. コンピュータのコマンド・プロンプトで、データベースを作成するディレクトリに移動します。

   次に例を示します。

   ```
   cd temp
   ```

2. 次のコマンドを入力して、データベースを作成します。

   ```
   dbinit -s database-name.db
   ```

   -s オプションは、データベースのチェックサムを有効にします。

   ヒント
   暗号化やページ・サイズなど、データベースのプロパティも初期化ユーティリティを使用して設定できます。「初期化ユーティリティ (dbinit)」 662 ページを参照してください。

3. Windows CE デバイスへのデータベースのコピー

   Windows CE デバイスにデータベースをコピーすることの詳細については、「Windows CE デバイスへのデータベースのコピー」 1051 ページを参照してください。

CREATE DATABASE 文を使用した Windows CE データベースの作成

CREATE DATABASE 文を使用して、Interactive SQL でコンピュータにデータベースを作成できます。ただし、このアプリケーションから Windows CE デバイスに直接データベースをコピーすることはできません。Windows CE デバイスにデータベースを手動でコピーする必要があります。

Windows CE データベース用にチェックサムを有効化
Windows CE 上に配備するデータベースを作成する場合は、チェックサムを有効にする必要があります。これによって、データベース・ファイルの破損を速やかに検出できます。

♦ **CREATE DATABASE 文を使用してデータベースを作成するには、次の手順に従います。**

1. コンピュータで Interactive SQL を開きます。

   [スタート] メニューから、[プログラム] – [SQL Anywhere 10] – [Interactive SQL] の順に選択します。

2. [接続] ダイアログが表示されます。

   [接続] ダイアログが自動的に表示されない場合は、[SQL] メニューから [接続] を選択してください。

3. [ID] タブで、[ODBC データ・ソース名] を選択し、隣接するフィールドで「**SQL Anywhere 10 Demo**」と入力します。

4. [OK] をクリックして、サンプル・データベースに接続します。

5. Interactive SQL の [SQL 文] ウィンドウ枠で、次の文を入力します。

   ```
   CREATE DATABASE 'c:¥¥temp¥¥database-name.db'
   TRANSACTION LOG ON
   CHECKSUM ON;
   ```

   ヒント
   暗号化やページ・サイズなど、データベースのプロパティも CREATE DATABASE 文を使用して設定できます。「CREATE DATABASE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

6. [SQL] – [実行] を選択します。

   コンピュータの *c:¥temp* ディレクトリに、データベースとトランザクション・ログが作成されます。

   データベースを Windows CE デバイスにコピーすることの詳細については、「Windows CE デバイスへのデータベースのコピー」 1051 ページを参照してください。

Windows CE デバイスへのデータベースのコピー

この項で説明する方法を使用すると、SQL Anywhere の既存のデータベースを Windows CE デバイスにコピーできます。ただし、Windows CE デバイスにデータベースをコピーした場合、Windows CE でサポートされていないデータベース機能は動作しないことに注意してください。「Windows CE での SQL Anywhere 機能のサポート」 1066 ページを参照してください。

♦ **データベースを Windows CE デバイスにコピーするには、次の手順に従います。**

1. Windows CE デバイスをコンピュータに接続します。

2. コンピュータで Windows エクスプローラを開きます。

3. コピーするデータベースが入っているディレクトリまで移動します。

4. データベース・ファイルを右クリックして、[コピー] を選択します。
5. Windows エクスプローラをもう 1 つ開きます。
6. データベース・ファイルを保存する Windows CE デバイスのディレクトリまで移動します。

> **ヒント**
> Windows CE デバイスで [サーバ起動オプション] ダイアログを使用してデータベースを開始するときに、参照ボタンを使用すると My Documents ディレクトリ内のデータベース・ファイルの検索しかできません。
> データベースを My Documents ディレクトリに保存していない場合は、[サーバ起動オプション] ダイアログの [データベース] フィールドにデータベースのパスを入力する必要があります。

7. Windows CE デバイスの [Windows エクスプローラ] ウィンドウで空白領域を右クリックして、[貼り付け] を選択します。

   ファイルが Windows CE デバイスにコピーされます。

Windows CE のデータベースの再構築

Windows CE のデータベースを再構築するには、以下のオプションがあります。

- 別のプラットフォームで Windows CE のデータベースを再構築し、Windows CE デバイスにデータベースをコピーする (これは、Windows CE データベースを再構築するときに推奨される方法です)
- dbmlsync を使用して空のデータベースを再配置する
- dbremote を使用して空のデータベースを再配置する
- Windows CE デバイス上で dbunload を使用する

最初の 3 つのオプションは、Windows CE データベースをアップグレードするときに推奨されます。これらのオプションを利用できない場合は、Windows CE 上で dbunload を使用できます。Windows CE 上で dbunload を使用することを決定する前に、Windows CE 上で dbunload を使用した場合に生じる次のような影響を検討する必要があります。

- データベース・サーバのテンポラリ・ファイルのサイズ (アンロードや再ロードを実行すると、ファイルのサイズが数メガバイトになることがあります)
- dbunload および関連コンポーネント用に必要となる、追加の領域
- Windows CE デバイスのデータベースを複数コピーすることによって発生する、追加のコスト

Windows CE デバイスで dbunload を実行すると、利用可能なデバイスよりも多くのリソースが必要になることがあるため、可能であれば、異なるプラットフォーム上でデータベースをアップグレードすることをおすすめします。

注意
Windows CE デバイス上で dbunload を実行する場合には、[SQL Anywhere 10 for Windows CE の配備] ウィザードの [アンロード／再ロードのサポート] オプションを選択する必要があります。SQL Anywhere for Windows CE を最初にインストールしたときにこのオプションを選択していない場合は、SQL Anywhere インストールを変更して、このサポートを追加できます。

Windows CE で dbunload を使用する場合の注意

Windows CE デバイスで dbunload を使用するには、以下のタスクを実行しておく必要があります。

- 以下のファイルを、SQL Anywhere インストール・ディレクトリに配備します (デフォルトでは、*¥Program Files¥SQLAny10*)。
  - *dbsrv10.exe*
  - *dbunlspt.exe*
  - *dbunload.exe*
  - *dbrunsql.exe*
  - *unloadold.sql*
  - *unload.sql*
  - *optdeflt.sql*
  - *opttemp.sql*
- 以下のファイルを *¥Windows* ディレクトリに配備します。
  - *dblgen10.dll*
  - *dblib10.dll*
  - *dbtool10.dll*
  - *dbusen.dll*
- レジストリ エントリ *HKEY_LOCAL_MACHINE¥SOFTWARE¥Sybase¥SQL Anywhere¥10.0¥Location* の文字列値を、SQL Anywhere ソフトウェア・ディレクトリに設定します。

以下の手順をサードパーティ製 Windows CE アプリケーションに組み込んで、エンド・ユーザに対する処理を自動化するようにできます。これを行うように選択する場合は、-qc と -q のいずれかまたは両方のオプションを指定した dbunload および dbrunsql を使用すること、または *dbtool10.dll* で DBUnload 関数を呼び出すことを検討してください。

♦ Windows CE のデータベースをアンロードするには、次の手順に従います (dbunload)。

1. Windows CE 以外のプラットフォームでは、新しい空の SQL Anywhere 10 データベースを作成します。

   CHAR 照合順は、既存のデータベースと一致する必要があります。NCHAR UCA ソートが必要ない場合は、NCHAR 照合順は UTF8BIN にします (この場合、データベース・サーバに *dbicu10.dll* は必要ありません)。

2. SQL Anywhere 10 ソフトウェアと空の SQL Anywhere 10 データベース・ファイルを、Windows CE デバイスにコピーします。「Windows CE で dbunload を使用する場合の注意」 1053 ページを参照してください。
3. デバイス上で実行されているデータベース・サーバがないことを確認します。
4. 次のコマンドを実行します。

   ```
   dbunload-path¥dbunload -c "UID=DBA;PWD=DBA-password;CHARSET=none;DBF=existing-database" unload-directory
   ```
5. dbunload が成功したことを確認したら、dbunload ウィンドウを閉じます。
6. 次のコマンドを実行します。

   ```
   dbrunsql-path¥dbrunsql -c "UID=DBA;PWD=sql;CHARSET=none;DBF=new-empty-SQLAnywhere10databasefile" -g- ¥reload.sql
   ```
7. dbrunsql が成功したことを確認したら、dbrunsql ウィンドウを閉じます。
8. Windows CE デバイスから *reload.sql* ファイルと *unload-directory* ファイルを削除します。

Windows CE データベースのバックアップ

データが破損したりメディアに障害が発生した場合にデータを失わないように、バックアップとリカバリが重要になります。Windows CE デバイスの盗難や紛失、またはメディアの障害によるデータ損失を防ぐには、Windows CE のデータベースは物理的に別の場所にバックアップするのが最善策です。

バックアップとリカバリのユーティリティのほとんどは、Windows CE で使用できます。ただし、Windows CE ではバックアップを物理的に別の場所に保存できないため、これらのユーティリティは役に立ちません。代わりに、データベース・ファイル全体をコンピュータにコピーしてデータをバックアップします。また、同期を使用して Windows CE のデータベースのコピーをコンピュータに維持することもできます。「Mobile Link 同期について」 『Mobile Link - クイック・スタート』を参照してください。

Windows CE データベースの消去

SQL Anywhere for Windows CE は、[データベース消去] ウィザード、DROP DATABASE 文、消去ユーティリティ (dberase) をサポートしていません。Windows CE デバイスからデータベースを手動で消去する必要があります。実行中のデータベースは削除できません。

Windows CE からデータベースを消去する方法は、2 通りあります。デバイスのインタフェースを使ってデータベースを消去するか、デバイスをコンピュータに接続して Windows エクスプローラを使ってデータベースを消去します。

データベースを削除したあと、トランザクション・ログ・ファイルを削除します (ファイルが存在する場合)。

♦ **デバイスのインタフェースを使用してデータベースを消去するには、次の手順に従います。**

1. [スタート] メニューから、[プログラム] – [ファイル エクスプローラ] の順にタップして、削除するデータベース・ファイルのあるディレクトリに移動します。
2. データベース・ファイルをタップして押したままにします。

   ポップアップ・メニューが表示されます。
3. [削除] をタップします。
4. [はい] をクリックして、削除を確認します。

♦ **Windows エクスプローラを使用してデータベースを消去するには、次の手順に従います。**

1. Windows CE デバイスをクレードルに置き、ActiveSync を介してコンピュータに接続していることを確認します。
2. コンピュータで Windows エクスプローラを開きます。
3. データベース・ファイルが格納されている Windows CE ディレクトリまで参照します。
4. データベース・ファイルを右クリックして、[削除] を選択します。

   削除を確認するプロンプトが表示された場合は、[はい] をクリックします。

Windows CE 上でのデータベース・サーバの実行

通常のクライアント／サーバの配置では、データベース・サーバは、クライアント・アプリケーションのコンピュータより高性能でリソースが多いコンピュータで稼働します。しかし、Windows CE ではこれは当てはまりません。Windows CE では、それほど性能が高くないコンピュータでデータベース・サーバを稼働します。

Windows CE には、ネットワーク・データベース・サーバが提供されています。ファイル名は *dbsrv10.exe* です。ネットワーク・データベース・サーバは、TCP/IP 経由の通信をサポートします。Windows CE はネットワーク・データベース・サーバをサポートするため、コンピュータで管理ユーティリティを使用して、Windows CE データベースに対するタスクを実行できます。次に例を示します。

- コンピュータで Sybase Central を使用してデータベースを管理できる
- コンピュータ上の Interactive SQL を使用して、データのロードとアンロード、クエリを実行できる

詳細については、「Windows CE での管理ユーティリティの使用」 1058 ページを参照してください。

Windows CE データベース・サーバは、次のコマンドによる明示的な要求がないかぎり、TCP/IP ネットワーク・リンクを開始しません。

Windows CE でのデータベース・サーバの起動については、「チュートリアル：Sybase Central から Windows CE データベースを実行する」 1058 ページを参照してください。

Windows CE では、すでに最初の SQL Anywhere データベース・サーバを実行中に 2 番目のサーバを起動しようとしても、最初のデータベース・サーバだけが動作します。Windows CE アプリケーションではこれが標準の動作になります。そのため、Windows CE デバイスでは 2 つのデータベース・サーバを同時に実行することはできません。ただし、SQL Anywhere は 1 つのデータベース・サーバ上で複数のデータベースを実行することはサポートします。

Windows CE でのサーバ・オプションの指定

SQL Anywhere の動作とパフォーマンスをチューニングするために、データベース・サーバを起動するときにサーバとデータベースを指定できます。さまざまなオプションを選択して、キャッシュで使用できるメモリ、データベース・サーバ上のデータベースを起動するのに必要なパーミッションのレベル、使用するネットワーク・プロトコルなどの機能を指定できます。

Windows CE では、オプションは [サーバ起動オプション] ダイアログで指定します。データベース・サーバのオプションをコマンド・ラインで設定する他の Windows オペレーティング・システムとは、この点で異なります。ほとんどのサーバ・オプションは、Windows CE でも使用できます。

データベース・サーバ・オプションの詳細については、「データベース・サーバ」 137 ページを参照してください。

サポートされていないオプションの詳細については、「Windows CE でサポートされているデータベース・サーバ・オプション」 1068 ページを参照してください。

Windows CE での管理ユーティリティの使用

この項では、Windows CE のデータベースで SQL Anywhere データベース管理ユーティリティを使用するときに考慮すべき点を説明します。

チュートリアル：Sybase Central から Windows CE データベースを実行する

Sybase Central は、SQL Anywhere を管理するためのグラフィカル・ユーザ・インタフェースを提供するデータベース管理ツールです。Sybase Central は、Mobile Link 同期など他の製品を管理する場合にも使用できます。

このチュートリアルを完了すると、サーバの開始と停止、データベース・サーバ上での単一または複数のデータベースの実行、データベースへの接続など、データベース・サーバに関連する主要タスクを実行できるようになります。

前提条件

- このチュートリアルを開始する前に、以下のすべてのタスクを完了します。
  - 「SQL Anywhere for Windows CE」 1027 ページ
  - 「Windows CE デバイスで実行中のデータベースへの接続」 1039 ページ
  - 「Windows CE デバイスに接続するための ODBC データ・ソースの作成」 1041 ページ
- Windows CE デバイスをコンピュータに接続します。

始める前に

チュートリアルで使用する Windows CE のデータベースを 2 つ作成する必要があります。

♦ Windows CE デバイスのデータベースを作成するには、次の手順に従います。

1. Windows CE デバイスが、コンピュータに接続していることを確認します。
2. [スタート] – [プログラム] – [SQL Anywhere 10] – [Sybase Central] を選択して、Sybase Central を開きます。

   Sybase Central が表示されます。
3. [ツール] – [SQL Anywhere 10] – [データベースの作成] を選択して、[データベース作成] ウィザードを起動します。

   [データベース作成] ウィザードが表示されます。
4. データベースの名前を **Alpha.db** に設定します。[次へ] をクリックします。

   コンピュータのコピーは、Windows CE デバイスにデータベースをコピーするときに削除されるので、コンピュータでデータベースを格納するディレクトリは無視できます。
5. [Windows CE にこのデータベースを作成] オプションを選択します。[次へ] をクリックします。

Windows CE デバイスとの接続がテストされます。

6. [データベースを Windows CE デバイスにコピー] オプションを選択します。

   [Windows CE ファイル名] フィールドに「*¥My Documents¥Alpha.db*」と入力して、Windows CE デバイスの My Documents ディレクトリにデータベースをコピーします。[次へ] をクリックします。

7. [コピー後にデスクトップのデータベースを削除] オプションを選択します。

8. ウィザードの残りの指示に従います。

9. [完了] をクリックすると、データベースがデフォルトの設定で作成されます。

   データベース・ファイル *Alpha.db* が Windows CE デバイスにコピーされたら、次の手順に進みます。

10. 3 ー 9 の手順を繰り返して、*Beta.db* というデータベースを作成します。

    2 番目のデータベースの手順 6 で入力するパスは、*¥My Documents¥Beta.db* です。

データベース・ファイル *Beta.db* が Windows CE デバイスにコピーされたら、チュートリアルを開始できます。

レッスン 1 ：データベース・サーバの起動

この項では、Windows CE で単一のデータベースを実行するという簡単な例を紹介します。

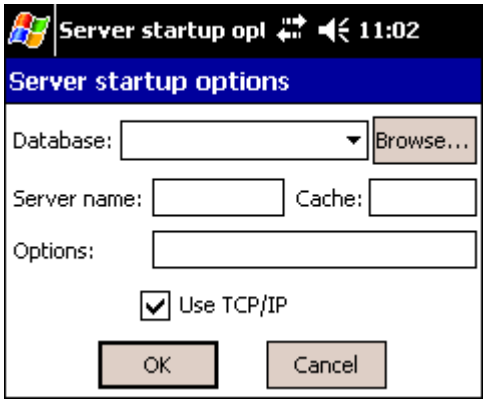
♦ **サーバでデータベースを開始するには、次の手順に従います。**

1. [スタート] メニューから、[プログラム] – [ファイル エクスプローラ] の順にタップして、Windows CE デバイス上でファイル・エクスプローラを起動します。

2. [プログラム] – [SQLAny10] の順にタップして、SQL Anywhere インストール・ディレクトリに移動します。

3. *dbsrv10* アイコンをタップします。

   [サーバ起動オプション] ダイアログが表示されます。



4. 開始するデータベース・ファイルを指定します。

   [参照] をタップして、*My Documents* ディレクトリの *Alpha.db* ファイルを見つけます。

5. [サーバ名] フィールドに「**CEserver**」と入力して、ネットワーク・データベース・サーバに名前を割り当てます。

6. デフォルトのキャッシュ・サイズ 600 KB を使用します。

   ヒント
   このチュートリアルの目的上、デフォルトのキャッシュ・サイズで十分です。大きいデータベースの場合は、キャッシュ・サイズを大きくするとパフォーマンスが向上します。「パフォーマンス向上へのキャッシュの使用」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

7. [TCP/IP を使用] オプションを選択します。

   コンピュータから Windows CE デバイスで実行中のデータベースに接続するには、TCP/IP 接続が必要です。後のレッスンで、コンピュータから接続します。

8. [サーバ起動オプション] ダイアログの [オプション] フィールドに、「**-gd all**」と入力します。

   -gd オプションでパーミッションを設定すると、どのユーザでもネットワーク・データベース・サーバで追加のデータベースを開始できます。これは、レッスンの後半で必要になります。「-gd サーバ・オプション」 168 ページを参照してください。

9. [OK] をタップして、ネットワーク・データベース・サーバで実行している Alpha データベースを開始します。

10. デバイスの [Today] 画面に移動します。

11. 画面の右下にあるデータベース・サーバ・アイコンをタップします。

    サーバ・メッセージ・ウィンドウが表示されます。

サーバ・メッセージ・ウィンドウに「**要求を受信中です。**」というメッセージが表示されたら、次のレッスンに進んでください。

次の作業

次は、Windows CE 上のネットワーク・データベース・サーバで複数のデータベースを開始する方法を学習します。

レッスン 2 ：Windows CE データベース・サーバで複数のデータベースを開始する

Windows CE デバイスでは、すでに最初の SQL Anywhere データベース・サーバを実行中に 2 番目のサーバを起動しようとしても、最初のデータベース・サーバだけが動作します。Windows CE アプリケーションではこれが標準の動作になります。このために Windows CE デバイスでは 2 つのデータベース・サーバを同時に実行することはできません。複数のデータベース・サーバを実行する代わりに、1 つのサーバで複数のデータベースを実行できます。

♦ Sybase Central からデータベースに接続するには、次の手順に従います。

1. [スタート] – [プログラム] – [SQL Anywhere 10] – [Sybase Central] を選択して、Sybase Central を起動します。

   Sybase Central が開始します。

2. [接続] メニューから [SQL Anywhere 10 に接続] を選択します。

   [接続] ダイアログが表示されます。

3. [ID] タブで、次の値を入力します。

   ♦ ユーザ ID　DBA

   ♦ パスワード　sql

4. [ODBC データ・ソース名] オプションを選択します。

5. [参照] をクリックし、「Windows CE デバイスに接続するための ODBC データ・ソースの作成」 1041 ページで作成した **CEdevice** データ・ソースを選択します。

6. [データベース] タブの [サーバ名] フィールドに、サーバ名 **CEserver** を入力します。

7. [OK] をクリックして、Windows CE デバイスで実行中の *Alpha.db* データベースに接続します。

データベース・サーバを起動して Alpha データベースに接続したら、Windows CE デバイスで他のデータベースを開始できます。

♦ ネットワーク・データベース・サーバで 2 番目のデータベースを開始するには、次の手順に従います。

1. コンテキスト・ドロップダウン・リストから、**CEserver** を選択します。

2. [ファイル] メニューから、[データベースの開始] を選択します。

   [データベースの開始] ダイアログが表示されます。

3. [データベース・ファイル] フィールドに、パス「**¥My Documents¥Beta.db**」を入力します。
4. [サーバ名] フィールドに、サーバ名「**CEserver** 」を入力します。
5. [OK] をクリックして、ネットワーク・データベース・サーバでデータベースを開始します。

ネットワーク・データベース・サーバにデータベースがロードされます。コンピュータから接続を開始します。

♦ **データベースに接続するには、次の手順に従います。**

1. Sybase Central の右ウィンドウ枠で、**Beta** データベース・アイコンを選択します。

   このアイコンは、Windows CE デバイスで先ほど開始したデータベースを表します。
2. [ファイル] メニューから、[接続] を選択します。

   [接続] ダイアログが表示されます。
3. [ID] タブで、次の値を入力します。

   ♦ **ユーザ ID　DBA**

   ♦ **パスワード　sql**
4. [データベース] タブで、[データベース・ファイル] フィールドにデータベース名「**¥My Documents¥Beta.db**」を入力します。
5. [OK] をクリックして、Windows CE デバイスで実行中の Beta データベースに接続します。

これで、Sybase Central を使って Alpha データベースと Beta データベースを表示して操作できます。

次の作業

次は、Windows CE でデータベースを切断してデータベース・サーバを停止する方法を学習します。

レッスン 3 ：Windows CE でデータベース・サーバを停止する

Windows CE デバイスでネットワーク・データベース・サーバを停止する前に、コンピュータから接続を切断する必要があります。

♦ **Windows CE のデータベースから接続を切断するには、次の手順に従います。**

1. Sybase Central から、[接続] – [切断] を選択します。

   [接続] ダイアログが表示されます。
2. Alpha データベースへの接続を選択します。
3. [切断] をクリックします。

4. 1 ー 3 の手順を繰り返して、Beta データベースへの接続を切断します。

Sybase Central の Windows CE データベースから切断したので、ネットワーク・データベース・サーバを停止できます。

♦ **サーバを停止するには、次の手順に従います。**

1. [Today] 画面の右下にあるデータベース・サーバ・アイコンをタップします。

   サーバ・メッセージ・ウィンドウが表示されます。

2. [シャットダウン] をタップします。

参考資料

Sybase Central からデータベースに接続したら、データベース内のテーブルへのデータ追加、データベース・オブジェクトの追加や編集、その他の管理タスクの実行、などを行うことができます。

Sybase Central からデータベースを管理することの詳細については、次の項を参照してください。

- 「SQL Anywhere プラグインの使用」 581 ページ
- 「データベースの編集」 『SQL Anywhere サーバ - SQL の使用法』

チュートリアル：Interactive SQL を使用した Windows CE データベースの管理

Interactive SQL は、データベース内のデータの変更や問い合わせと、データベース構造の修正ができるアプリケーションです。Interactive SQL では、SQL 文を入力するためのウィンドウ枠が表示されます。また、クエリの進捗情報や結果セットを表示するウィンドウ枠も表示されます。

このチュートリアルでは、コンピュータから Interactive SQL を使用して Windows CE デバイス上のデータベースを管理する方法を簡単に紹介します。まず、Interactive SQL から Windows CE デバイスのサンプル・データベースに接続する方法を学習します。接続したら、Interactive SQL を使用して SQL 文を実行できます。

レッスン 1 ：サンプル・データベースの開始

サンプル・データベースは、Interactive SQL からの接続を行う前に、Windows CE デバイスで実行されていることが必要です。

♦ **サンプル・データベースを開始するには、次の手順に従います。**

1. [スタート] メニューから、[プログラム] – [ファイル エクスプローラ] の順にタップして、Windows CE デバイス上でファイル・エクスプローラを起動します。
2. [プログラム] – [SQLAny10] の順にタップして、SQL Anywhere インストール・ディレクトリに移動します。
3. dbsrv10 アイコンをタップします。

[サーバ起動オプション] ダイアログが表示されます。

4. [データベース] フィールドに、サンプル・データベースのパスを入力します。デフォルトのロケーションは **¥Program Files¥SQLAny10¥demo.db** です。ソフトウェアを別のローションにインストールする場合は、[参照] ボタンを使用して、データベースを検索します。
5. [サーバ名] フィールドに「**CEserver**」と入力して、ネットワーク・データベース・サーバに名前を割り当てます。
6. [キャッシュ] フィールドに「**5MB**」と入力して、キャッシュ・サイズを設定します。

   Windows CE のデフォルトのキャッシュ・サイズは 600 KB ですが、パフォーマンスを向上させるには、これより大きいキャッシュ・サイズをおすすめします。
7. [TCP/IP を使用] を選択します。このチュートリアルでは、デフォルトのプロキシ・ポート番号 2638 を使用することを前提にしています。
8. [OK] をクリックして、ネットワーク・サーバで実行中のサンプル・データベース・サーバを開始します。

   サーバ・ウィンドウが表示されて、すぐに閉じます。
9. デバイスの [Today] 画面に移動します。
10. 画面の右下にあるサーバ・アイコンをタップします。

    サーバ・メッセージ・ウィンドウが表示されます。

    サーバ・メッセージ・ウィンドウに「**要求を受信中です**」というメッセージが表示されたら、次のレッスンに進んでください。

次の作業

次は、Interactive SQL から Windows CE デバイスで実行中のデータベースに接続する方法を学習します。

レッスン 2 ：Interactive SQL を開始して接続する

サンプル・データベースが Windows CE デバイスで実行していれば、Interactive SQL から接続して、コンピュータからデータベースを表示して管理できます。

♦ Interactive SQL から Windows CE デバイスのデータベースに接続するには、次の手順に従います。

1. [スタート] – [プログラム] – [SQL Anywhere 10] – [Interactive SQL] を選択して、Interactive SQL を起動します。

   Interactive SQL が起動すると、[接続] ダイアログが自動的に表示されます。
2. [ID] タブで、次の値を入力します。

   ♦ **ユーザ ID　DBA**

- **パスワード　sql**

3. [ODBC データ・ソース名] オプションを選択します。
4. [参照] をクリックし、「Windows CE デバイスに接続するための ODBC データ・ソースの作成」 1041 ページで作成した **CEdevice** データ・ソースを選択します。

   [OK] をクリックします。
5. [データベース] タブの [サーバ名] フィールドに、サーバ名を入力します。

   この例では、前のレッスンで指定した CEserver がサーバ名です。
6. [OK] をクリックして、Windows CE デバイスで実行しているサンプル・データベースに接続します。

次の作業

これで、Interactive SQL からサンプル・データベースのデータを表示して管理できます。

レッスン 3 ： Windows CE データベースに対してクエリを実行する

Interactive SQL の重要な使用目的の 1 つは、テーブル・データをブラウズすることです。Interactive SQL では、データベース・サーバに要求を送信することによって情報を検索します。一方、データベースサーバは、情報を調べ、それを Interactive SQL に返します。

- **Windows CE データベースに対して SQL 文を実行するには、次の手順に従います。**

1. [SQL 文] ウィンドウ枠に次のように入力します。

   ```
   SELECT * FROM Employees
   ```
2. [SQL 文の実行] をクリックする、[SQL] – [実行] を選択する、[F5] キーを押す、のいずれかの操作を行って、文を実行します。

   Employees テーブルのすべてのデータが、[結果] ウィンドウ枠に表示されます。
3. [SQL] – [切断] を選択するか、[F12] キーを押して、Windows CE データベースから切断します。

参考資料

Interactive SQL からデータベースに接続すると、データベース・オブジェクトの追加や変更に加えて、データの表示や操作を行うことができます。

Interactive SQL、クエリの記述、SQL 文の使用の詳細については、次の項を参照してください。

- 「Interactive SQL」 585 ページ
- 「クエリ：テーブルからのデータの選択」 『SQL Anywhere サーバ - SQL の使用法』
- 「クエリ結果の要約、グループ化、ソート」 『SQL Anywhere サーバ - SQL の使用法』
- 「ジョイン：複数テーブルからのデータ検索」 『SQL Anywhere サーバ - SQL の使用法』
- 「SQL 文」 『SQL Anywhere サーバ - SQL リファレンス』

Windows CE での SQL Anywhere 機能のサポート

この項では、SQL Anywhere のコンポーネントや機能のうち、Windows CE でサポートされていないか、動作が異なるものをリストにします。サポートされていない機能の代わりになる機能がある場合は、それについても説明します。

Windows CE でサポートされるコンポーネントとサポートされないコンポーネントの詳細については、「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」を参照してください。

SQL Anywhere には、データベースを管理するための数々のツールが付属しています。たとえば、Sybase Central、Interactive SQL、コマンド・ライン・ユーティリティなどがあります。これらの管理ツールは、Windows CE では使用できません。データベースの管理は、Windows CE デバイスに接続している Windows ベースのコンピュータから行います。

詳細については、「Windows CE での管理ユーティリティの使用」 1058 ページを参照してください。

コンポーネントまたは機能	考慮事項
アプリケーション・プロファイリング	Windows CE で実行しているデータベースのトレーシング・セッションを作成する場合は、[データベース・トレーシング] ウィザードを使用してトレースを設定する必要があります。[アプリケーション・プロファイリング] ウィザードは使用できません。また、Windows CE デバイスのデータをトレースし、デスクトップ・コンピュータ上のデータベース・サーバで実行している Windows CE データベースのコピーに格納する必要があります。Windows CE デバイスからトレーシング・データベースを自動的に作成することはできません。また、CE デバイス上のローカル・データベースにトレースすることはできません。「アプリケーション・プロファイリング」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
データベース・ミラーリング	Windows CE ではサポートされていません。
外部ストアド・プロシージャ	Windows CE ではサポートされていません。
iAnywhere JDBC ドライバ	Windows CE ではサポートされていません。 Windows CE 上で jConnect を使用できます。
データベースの Java	Windows CE ではサポートされていません。

コンポーネントまたは機能	考慮事項
jConnect	jConnect ドライバは、Windows CE のデータベースを作成するときに有効にできます。これは、Java をサポートしているコンピュータにデータベースを移動する場合に便利です。ただし、データベース・サイズが大きくなり、jConnect 機能を使用しない場合でも、データベースを実行するための必要メモリが約 200 KB 増えます。Windows CE の限られたメモリ環境でデータベースを実行する場合は、この追加のメモリ要件を考慮してください。
Kerberos 認証	Windows CE ではサポートされていません。
LDAP 認証	Windows CE ではサポートされていません。
ODBC クライアント	Windows CE では、ODBC ドライバ・マネージャまたは ODBC アドミニストレータを提供していないため、SQL Anywhere はファイルに保管されている ODBC データ・ソースを使用します。「Windows CE での ODBC データ・ソースの使用」 80 ページを参照してください。
Open Client	Windows CE ではサポートされていません。
並列バックアップ	Windows CE ではサポートされていません。
パーソナル・データベース・サーバ (dbeng10)	Windows CE でサポートされているのはネットワーク・データベース・サーバ (*dbsrv10*) のみです。この実行プログラムは、ローカル接続とネットワーク経由のクライアント／サーバ通信をサポートしています。
リモート・データ・アクセス (ディレクトリ・アクセス・サーバを含む)	Windows CE ではサポートされていません。
SPX プロトコル	Windows CE は、TCP/IP プロトコルを使用します。

Windows CE での SQL 文のサポート

この項では、Windows CE でサポートされていないか、機能が異なるか、機能に制限のある SQL 文について説明します。

SQL 文の全リストについては、「SQL 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

SQL 文	考慮事項
BACKUP 文	Windows CE でサポートされているのは、BACKUP DATABASE DIRECTORY 句のみです。

SQL 文	考慮事項
CREATE DATABASE 文	CREATE DATABASE 文はコンピュータのデータベースを初期化できるため、これをあとで Windows CE デバイスにコピーできます。「Windows CE データベースの作成」 1047 ページを参照してください。
CREATE EVENT 文	DiskSpace イベント・タイプは、Windows CE ではサポートされていません。ただし、この文は GlobalAutoIncrement または ServerIdle イベント・タイプの定義に使用できます。
CREATE EXISTING TABLE 文	Windows CE ではサポートされていません。
CREATE EXTERNLOGIN 文	Windows CE ではサポートされていません。
CREATE FUNCTION 文	データベースでユーザ定義の SQL 関数を作成する場合は、Windows CE でも CREATE FUNCTION 文を使うことができます。Windows CE では EXTERNAL NAME 句はサポートされていないことに注意してください。
CREATE SERVER 文	Windows CE ではサポートされていません。
CREATE TABLE 文	プロキシ・テーブルを作成するための CREATE TABLE 文の AT 句は、Windows CE ではサポートされていません。
DROP DATABASE 文	Windows CE ではサポートされていません。
DROP SERVER 文	Windows CE ではサポートされていません。
INSTALL JAVA 文	Windows CE ではサポートされていません。
REMOVE JAVA 文	Windows CE ではサポートされていません。
REORGANIZE TABLE 文	Windows CE ではサポートされていません。
RESTORE DATABASE 文	Windows CE ではサポートされていません。
START JAVA 文	Windows CE ではサポートされていません。
STOP JAVA 文	Windows CE ではサポートされていません。

Windows CE でサポートされているデータベース・サーバ・オプション

この項では、Windows CE でサポートされていないか、機能が異なるデータベース・サーバ・オプションについて説明します。

オプション	考慮事項
@data オプション	Windows CE ではサポートされていません。
-? サーバ・オプション	Windows CE ではサポートされていません。
-cm サーバ・オプション	Windows CE ではサポートされていません。

オプション	考慮事項
-cw サーバ・オプション	Windows CE ではサポートされていません。
-ec サーバ・オプション	強力な通信暗号化は、Windows CE ではサポートされていません。**none** (なし)と **simple** (単純) の設定だけがサポートされています。「-ec サーバ・オプション」 161 ページを参照してください。
-gb サーバ・オプション	Windows CE ではサポートされていません。
-ge サーバ・オプション	Windows CE ではサポートされていません。
-gss サーバ・オプション	Windows CE ではサポートされていません。
-qi サーバ・オプション	実行中、ネットワーク・データベース・サーバは Windows CE デバイスの [Today] 画面の右下にアイコンとして表示されます。この機能は無効にできません。
-s サーバ・オプション	Windows CE ではサポートされていません。
-tmf サーバ・オプション	Windows CE ではサポートされていません。
-tmt サーバ・オプション	Windows CE ではサポートされていません。
-u サーバ・オプション	Windows CE ではサポートされていません。
-ua サーバ・オプション	Windows CE ではサポートされていません。
-uc サーバ・オプション	Windows CE ではサポートされていません。
-ud サーバ・オプション	Windows CE ではサポートされていません。
-uf サーバ・オプション	Windows CE ではサポートされていません。
-ui サーバ・オプション	Windows CE ではサポートされていません。
-ut サーバ・オプション	Windows CE ではサポートされていません。
-ux サーバ・オプション	Windows CE ではサポートされていません。
-xp サーバ・オプション	Windows CE ではサポートされていません。

Windows CE での Sybase Central ウィザードのサポート

次の表に、Windows CE でサポートされていない Sybase Central のウィザード、機能が異なるウィザード、その代わりとなる方法 (可能な場合は) をリストします。

ウィザード	考慮事項
[データベース・バックアップ] ウィザード	アーカイブ・バックアップは、Windows CE ではサポートされていません。したがって、[データベース・バックアップ] ウィザードはサポートされていません。「バックアップの種類」 819 ページを参照してください。 別の方法として、Windows CE では、[バックアップ・イメージ作成] ウィザードを使用できます。このウィザードは、データベース・ファイルとトランザクション・ログ・ファイルのバックアップをそれぞれ別のファイルとして作成します。「バックアップを作成し、元のトランザクション・ログを継続して使用する」 849 ページを参照してください。
[ログ・ファイル設定の変更] ウィザード	Windows CE ではサポートされていません。
[データベース作成] ウィザード	このウィザードは、Windows CE にデータベースを作成するオプションを提供します。この場合、Sybase Central を実行しているコンピュータ上に Windows CE サービスがインストールされていることが前提となります。「Windows CE データベースの作成」 1047 ページを参照してください。
[メンテナンス・プラン作成] ウィザード	以下のオプションは、Windows CE では利用できません。 ◆ フル・アーカイブ・バックアップ ◆ テープにバックアップ ◆ メンテナンス・プラン・レポートを電子メールで送信する
[データベース消去] ウィザード	Windows CE ではサポートされていません。
[データベース移行] ウィザード	Windows CE ではサポートされていません。
[データベース・リストア] ウィザード	Windows CE ではサポートされていません。
[サービス作成] ウィザード	Windows CE ではサポートされていません。
[ログ・ファイル変換] ウィザード	Windows CE ではサポートされていません。
[データベース・アンロード] ウィザード	このウィザードは、データベース・ファイルが格納されている Windows CE ディレクトリにはマップできません。ただし、Windows CE のデータベースをコンピュータにコピーすると、[データベース・アンロード] ウィザードを使用してアンロードできます。

ウィザード	考慮事項
[データベース・アップグレード] ウィザード	このウィザードは、Windows CE ではサポートされていません。ただし、Windows CE のデータベースをコンピュータにコピーし、このウィザードを使ってから Windows CE デバイスにコピーし直すと、アップグレードできます。「データベースのアップグレード」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

Windows CE での SQL Remote のサポート

SQL Remote は、次の例外を除いて Windows CE ではサポートされていません。

コンポーネントまたは機能	考慮事項
抽出ユーティリティ (dbxtract)	Windows CE は、このユーティリティをサポートしていません。必要な場合は、Windows CE のデータベースをコンピュータにコピーすると、抽出ユーティリティを使えるようになります。
MAPI メッセージ・タイプ	このメッセージ・システムは、Windows CE ではサポートされていません。
VIM メッセージ・タイプ	このメッセージ・システムは、Windows CE ではサポートされていません。

索引

記号

A

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

あ

い

う

え

お

か

き

け

こ

さ

し

す

せ

そ

た

ち

と

な

に

ね

は

ひ

ふ

へ

ほ

ま

み

め

も

ゆ

よ

ら

り

る

れ

ろ

