iAnywhere SOLUTIONS™
A SYBASE COMPANY

Adaptive Server® Anywhere
データベース管理ガイド

パート番号：DC03927-01-0902-01
改訂：2005 年 3 月

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、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他に、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、セキュリティ、バックアップ・プロシージャ、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、Sybase Adaptive Server 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 Studio 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 Studio のマニュアル

SQL Anywhere Studio のマニュアルは、各マニュアルを 1 つの大きなヘルプ・ファイルにまとめたオンライン形式、マニュアル別の PDF ファイル、および有料の製本版マニュアルで提供されます。SQL Anywhere Studio のマニュアルは、次の分冊マニュアルで構成されています。

- **『SQL Anywhere Studio の紹介』** このマニュアルでは、SQL Anywhere Studio のデータベース管理と同期テクノロジの概要について説明します。また、SQL Anywhere Studio を構成する各部分について説明するチュートリアルも含まれています。
- **『SQL Anywhere Studio 新機能ガイド』** このマニュアルは、SQL Anywhere Studio のこれまでのリリースのユーザを対象としています。ここでは、製品の今回のリリースと以前のリリースで導入された新機能をリストし、アップグレード手順を説明しています。
- **『Adaptive Server Anywhere データベース管理ガイド』** このマニュアルでは、データベースおよびデータベース・サーバの実行、管理、設定について説明しています。
- **『Adaptive Server Anywhere SQL ユーザーズ・ガイド』** 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。
- **『Adaptive Server Anywhere SQL リファレンス・マニュアル』** このマニュアルは、Adaptive Server Anywhere で使用する SQL 言語の完全なリファレンスです。また、Adaptive Server Anywhere のシステム・テーブルとシステム・プロシージャについても説明しています。
- **『Adaptive Server Anywhere プログラミング・ガイド』** このマニュアルでは、C、C++、Java プログラミング言語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。また、Adaptive Server Anywhere ADO.NET データ・プロバイダについても説明します。

- **『Adaptive Server Anywhere SNMP Extension Agent ユーザーズ・ガイド』** このマニュアルは、Adaptive Server Anywhere SNMP Extension Agent を SNMP 管理アプリケーションで使用して Adaptive Server Anywhere データベースを管理するために設定する方法について説明します。

- **『Adaptive Server Anywhere エラー・メッセージ』** このマニュアルでは、Adaptive Server 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。

- **『SQL Anywhere Studio セキュリティ・ガイド』** このマニュアルでは、Adaptive Server Anywhere データベースのセキュリティ機能について説明します。Adaptive Server Anywhere 7.0 は、米国政府から TCSEC (Trusted Computer System Evaluation Criteria) の C2 セキュリティ評価を授与されています。このマニュアルには、Adaptive Server Anywhere の現在のバージョンを、C2 基準を満たした環境と同等の方法で実行することを望んでいるユーザにとって役に立つ情報が含まれています。

- **『Mobile Link 管理ガイド』** このマニュアルでは、モバイル・コンピューティング用の Mobile Link データ同期システムについてあらゆる角度から説明します。このシステムによって、Oracle、Sybase、Microsoft、IBM の単一データベースと、Adaptive Server Anywhere や Ultra Light の複数データベースの間でのデータ共有が可能になります。

- **『Mobile Link クライアント』** このマニュアルでは、Adaptive Server Anywhere リモート・データベースと Ultra Light リモート・データベースの設定を行い、これらを同期させる方法について説明します。

- **『Mobile Link サーバ起動同期ユーザーズ・ガイド』** 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の開始を可能にする Mobile Link の機能です。

- **『Mobile Link チュートリアル』** このマニュアルには、Mobile Link アプリケーションを設定して実行する方法を順を追って説明する複数のチュートリアルが含まれます。
- **『QAnywhere ユーザーズ・ガイド』** このマニュアルでは、Mobile Link QAnywhere について説明します。Mobile Link QAnywhere は、従来のデスクトップ・クライアントやラップトップ・クライアントだけでなく、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・アプリケーションの開発と展開を可能にするメッセージング・プラットフォームです。
- **『Mobile Link およびリモート・データ・アクセスの ODBC ドライバ』** このマニュアルでは、Mobile Link 同期サーバから、または Adaptive Server Anywhere リモート・データ・アクセスによって、Adaptive Server Anywhere 以外の統合データベースにアクセスするための ODBC ドライバの設定方法について説明します。
- **『SQL Remote ユーザーズ・ガイド』** 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、あらゆる角度から説明します。このシステムによって、Adaptive Server Anywhere または Adaptive Server Enterprise の単一データベースと Adaptive Server Anywhere の複数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 Studio ヘルプ』** このマニュアルには、Sybase Central や Interactive SQL、その他のグラフィカル・ツールに関するコンテキスト別のヘルプが含まれています。これは、製本版マニュアル・セットには含まれていません。
- **『Ultra Light データベース・ユーザーズ・ガイド』** このマニュアルは、Ultra Light 開発者を対象としています。ここでは、Ultra Light データベース・システムの概要について説明します。また、すべての Ultra Light プログラミング・インタフェースに共通する情報を提供します。
- **Ultra Light のインタフェースに関するマニュアル** 各 Ultra Light プログラミング・インタフェースには、それぞれに対応するマニュアルを用意しています。これらのインタフェースは、RAD (

ラピッド・アプリケーション開発) 用の Ultra Light コンポーネントとして提供されているものと、C、C++、Java 開発用の静的インタフェースとして提供されているものがあります。

このマニュアル・セットの他に、PowerDesigner と InfoMaker には、独自のオンライン・マニュアル (英語版) がそれぞれ用意されています。

マニュアルの形式

SQL Anywhere Studio 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 Studio 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] ー [プログラム] ー [SQL Anywhere 9] ー [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・ディレクトリに保存されている HTML マニュアルを参照してください。

- **PDF 版マニュアル** SQL Anywhere の各マニュアルは、Adobe Acrobat Reader で表示できる PDF ファイルで提供されています。

  PDF 版マニュアルは、オンライン・マニュアルまたは Windows の [スタート] メニューから利用できます。

- **製本版マニュアル** 製本版マニュアルをご希望の方は、ご購入いただいた販売代理店または弊社営業担当までご連絡ください。

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 つだけ選択するか、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、どちら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** }]

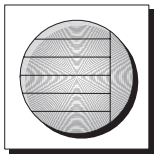
  QUOTES オプションを使用する場合は、ON または OFF のどちらかを選択する必要があります。角カッコと中カッコは入力しないでください。

グラフィック・アイコン

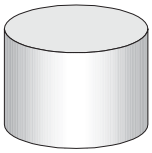
このマニュアルでは、次のアイコンを使用します。

- クライアント・アプリケーション

- Sybase Adaptive Server Anywhere などのデータベース・サーバ



- 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 同期サーバ、SQL Remote Message Agent などがあげられます。

- プログラミング・インタフェース

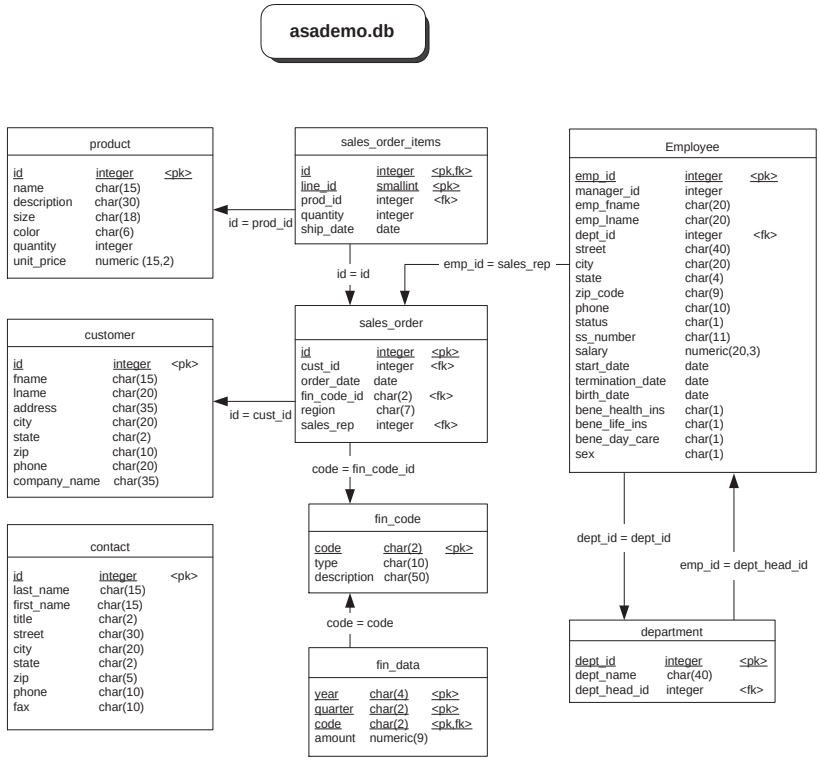
Adaptive Server Anywhere サンプル・データベース

このマニュアルでは、多くの例で Adaptive Server Anywhere サンプル・データベースが使用されています。

サンプル・データベースは、*asademo.db* という名前のファイルに保存され、SQL Anywhere ディレクトリに置かれています。

サンプル・データベースは小規模の企業の例です。データベースには、この企業の内部情報 (従業員、部署、経理) とともに、製品情報や販売情報 (受注、顧客、連絡先) が入っています。データベースに含まれる情報はすべて架空のものです。

次の図は、サンプル・データベース内のテーブルと各テーブル間の関係を示しています。



詳細情報の検索／フィードバックの提供

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。

マニュアルおよびソフトウェアに関するフィードバックは、SQL Anywhere のテクノロジについて議論するニュースグループを介してお送りいただけます。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 Solutions 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。

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する責任は負いませんが、お寄せ頂いたご意見、ご提案は必ず読ませて頂きます。

第 1 部　データベースの開始とデータベースへの接続

第 1 部では、Adaptive Server 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

データベース・サーバの実行

この章の内容

この章では、Adaptive Server Anywhere データベース・サーバの起動と停止方法、および起動するときに使用できるオプションについて、オペレーティング・システムごとに説明します。

概要

Adaptive Server Anywhere には、2 つのバージョンのデータベース・サーバが入っています。

- **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、単一ユーザによる同一マシン上での (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 *dbeng9.exe* です。Windows CE は、パーソナル・サーバをサポートしません。UNIX オペレーティング・システムでは、*dbeng9* 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 *dbsrv9.exe* です。Novell NetWare では *dbsrv9.nlm*、UNIX オペレーティング・システムでは *dbsrv9* です。

  パフォーマンスと信頼性の向上を確保するために、ネットワーク・データベース・サーバは Windows 95/98/Me ではなく Windows NT/2000/XP または Windows Server 2003 で実行することをおすすめします。

サーバ間の相違点

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。

- **CPU 数**　per-seat ライセンスの場合、ネットワーク・データベース・サーバはマシン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。

- **デフォルトの内部スレッド数**　-gn オプションを使用して、サーバが一度に処理できる要求の数を設定できます。パーソナル・データベース・サーバおよびネットワーク・データベース・サーバのデフォルトのスレッド数は 20 個ですが、Windows CE の場合はデフォルトは 3 個です。

  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」154 ページを参照してください。

- **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。

第一段階

パーソナル・サーバを単一のデータベースで起動するのは非常に簡単です。たとえば、*test.db* があるディレクトリで次のコマンドを入力すると、パーソナル・サーバとデータベース *test.db* の両方を起動できます。

```
dbeng9 test
```

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。

  詳細については、「StartLine 接続パラメータ [START]」273 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。

接続文字列でデータベース・ファイル名を使用して、パーソナル・サーバを起動することもできます。

詳細については、「組み込みデータベースへの接続」62 ページを参照してください。

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

- 特に指定のない限り、これらのコマンドは、パーソナル・サーバ (**dbeng9**) を起動します。ネットワーク・サーバを起動する場合は、**dbeng9** を **dbsrv9** 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* にファイル拡張子を指定していない場合、拡張子は *.db* とみなされます。

❖ 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。

```
start dbeng9 path¥database-file
```

データベース・ファイルを省略した場合は、[参照] ボタン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 上のデータベース・サーバの起動」10 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。

```
dbeng9 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) (*dbsrv9.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 dbsrv9.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・マシンからサーバをロードできます。詳細については、Novell のマニュアルを参照してください。

Novell *autoexec.ncf* ファイルにコマンドを入れておくと、NetWare サーバを起動するたびに、Adaptive Server 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・マシン上)、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、Adaptive Server 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。

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、[参照] ボタンを使用するとデータベース・ファイルを検索できるダイアログ・ボックスが表示されます。

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・システムでのファイル名については、「概要」4 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・サーバと同じ物理マシンに保存してください。ネットワーク・ドライブにデータベース・ファイルやトランザクション・ログを配置すると、パフォーマンスが低下したりデータが破壊されたりする可能性があります。

- **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。

各オプションの完全なリファレンス情報については、「データベース・サーバ」154 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。

```
dbeng9 -?
```

Windows CE 上のデータベース・サーバの起動

Windows CE で提供されるデータベース・サーバは、ネットワーク・データベース・サーバ (*dbsrv9.exe*) です。ネットワーク・サーバは、TCP/IP ネットワーク・リンクを介した通信をサポートしています。

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マシンより性能が高く多くのリソースを持つマシンで稼働します。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マシンでデータベース・サーバを稼働します。

Windows CE でネットワーク・サーバを使用すると、デスクトップ・コンピュータでデータベース・アプリケーションを稼働し、Windows CE データベースに対してタスクを実行できるという利点があります。次に例を示します。

- デスクトップ PC で Sybase Central を使用してデータベースを管理できる。
- デスクトップ上の 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。
- InfoMaker を使用してレポートを作成できる。

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、『SQL Anywhere Studio の紹介』>「レッスン 1 ：データベース・サーバを起動する」を参照してください。

Windows CE では、すでに最初の Adaptive Server 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。複数のサーバを実行する代わりに、必要に応じて 1 つのデータベース・サーバが複数のデータベースを実行できます。実行中のサーバでデータベースを自動的に起動するには、**-gd all** を使用する必要があることに注意してください。

-gd オプションの詳細については、「-gd サーバ・オプション」190 ページを参照してください。

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することができます。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
 path¥asademo.db
```

この例で、*path* は SQL Anywhere 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。

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。

```
dbeng9 @sample.cfg
```

詳細については、「@data サーバ・オプション」164 ページと「設定ファイルの使用」642 ページを参照してください。

サーバとデータベースの命名

–n は、サーバ・オプション (サーバの指定) またはデータベース・オプション (データベースの指定) 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、デスクトップ・アイコンとサーバ・ウィンドウのタイトル・バーに表示されます。

次のようなものは、データベース・サーバとデータベースの名前には使用できません。

- 空白スペースまたは一重か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

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、**asademo** 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。

```
dbeng9 -n Cambridge asademo.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。

```
dbeng9 -n Galt
```

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」25 ページを参照してください。

データベースの命名

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 **asademo**、次のデータベース名は **sample** です。

```
dbeng9 asademo.db sample.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。

```
dbeng9 asademo.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。

詳細については、「接続文字列と文字セット」448 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、Adaptive Server 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 と UNIX のオペレーティング・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。

パフォーマンス・チューニングの詳細については、『ASA SQL ユーザーズ・ガイド』>「パフォーマンスのモニタリングと改善」を参照してください。

キャッシュ・サイズの管理については、「キャッシュ・サイズ」161 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・サーバを使用しているマルチプロセッサ・マシンで実行している場合は、-gt オプションを使用してプロセッサ数を設定できます。

  詳細については、「-gt サーバ・オプション」197 ページと「コマンド・ラインからスレッドを制御する」17 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。

  起動オプションの詳細については、「データベース・サーバ」154 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。

詳細については、『SQL Anywhere Studio セキュリティ・ガイド』>「安全な方法でのデータベース・サーバの実行」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、8 MB のキャッシュを予約し、最大 4096 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。

```
dbsrv9 -gp 4096 -c 8M -n myserver
```

特殊モードでの実行

特定の目的のために、Adaptive Server Anywhere を特殊モードで実行することができます。

- **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。

詳細については、「-r サーバ・オプション」206 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。

  詳細については、「-b サーバ・オプション」166 ページと『ASA SQL ユーザーズ・ガイド』>「データのインポートとエクスポート」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。

  詳細については、「データベース・サーバ」154 ページを参照してください。

コマンド・ラインからスレッドを制御する

Adaptive Server Anywhere では、オペレーティング・システムの複数のスレッドを使用して要求を処理できます。これによって、異なる要求を別々の CPU で同時に実行できます。各要求は単一のスレッドで実行され、複数の CPU で同時に実行されるわけではありません。

スレッドを制御するオプションのリストについては、「スレッド動作の制御」19 ページを参照してください。

Adaptive Server Anywhere でのスレッド

スレッド・サポートの仕組みを理解するには、「**要求**」と「**タスク**」を理解する必要があります。

要求

Adaptive Server Anywhere が 2 人のユーザによって同時に使用されるか、1 つのアプリケーションから 2 つの接続が同時に行われるとします。それぞれの接続で、クエリ (または他の SQL 文) が Adaptive Server Anywhere に送信されます。送信される SQL 文は、それぞれサーバに対する個別の要求になります。

タスク

タスクは、1 つの要求を選択し、要求が完了するまでそれを処理します。Windows NT/2000/XP では、タスクはファイバと呼ばれる軽量スレッドです。他のすべてのプラットフォームでは、タスクはスレッドです。Adaptive Server Anywhere が同時に処理できる要求の数は、タスクの数によって決定します。

Windows 95/98/Me、NetWare、UNIX でのタスク

Windows 95/98/Me、NetWare、UNIX マシンでは、タスクはそれぞれがオペレーティング・システムのスレッドです。Adaptive Server Anywhere が要求を受信すると、オペレーティング・システム・スレッドがその要求を選択します。オペレーティング・システム・スレッドは、要求が完了するまで実行します。

最初の要求を実行中にサーバが次の要求を受信した場合は、2 番目の要求には別のスレッドが割り当てられます。スレッドがタスクの処理中にブロックされた場合、スレッドは別の要求を選択せず、現在実行中の要求を完了できるようになるまで待機します。

Windows NT/XP/2000 でのタスク

Windows NT/2000/XP では、タスクはファイバと呼ばれる軽量スレッドです。Adaptive Server Anywhere が要求を受信すると、ファイバがその要求を選択し、それが完了するまで実行します。最初の要求を実行中にデータベース・サーバが次の要求を受信した場合、2 番目の要求には別のファイバが割り当てられます。ファイバが要求の処理中にブロックした場合、ファイバが処理を進行できる状態になるまで、他のファイバに制御を譲ります。別の要求を選択することはありません。

スレッドはファイバの制御を司り、複数のファイバは協調性を持って動作するようにスケジュールされます。ファイバは、要求を処理するために待機している (たとえば I/O 操作の完了を待っている) ときは、制御を別のファイバに明示的に譲ります。ファイバがブロックしたが制御を譲らなかった場合、そのファイバをホスト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ホストしている場合、待機中のファイバをホスト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。

-gx オプションを使用して、ファイバを実行するために使用するスレッドの数を設定できます。Java またはリモート・データ・アクセスを使用していないかぎり、各 CPU に必要なスレッドは 1 つだけです。

Java またはリモート・データ・アクセスを使用しているときのスレッド数の設定

Java またはリモート・データ・アクセスを使用しているときに Java またはリモート・データ・アクセスを実行中のファイバがブロックした場合、そのファイバが別のファイバに制御を譲らず、スレッドをブロックすることがあります。このため、データベース・サーバに割り当てられるオペレーティング・システム・スレッドの数は、デフォルトでは、マシン上の CPU の数に 1 を足したものになっています。これにより、Java またはリモート・データ・アクセスによって使用されているスレッドがブロックされた場合でも、ファイバをホストするために使用できるスレッドが最低 1 つは確保されます。-gx オプションを使用してデフォルトより大きいスレッド数を指定しても、パフォーマンスにはほとんど影響しません。

スレッド動作の制御

スレッドの動作を制御するデータベース・サーバのオプションは 4 つあります。すべてのプラットフォームで、これらのオプションがすべて必要なわけではありません。

- **タスクの数**　-gn オプションは、要求を処理するために使用するタスクの数を制御します。この数値は、事実上は同時に処理できる要求の数になります。1 つの要求が 1 つのタスクを使用します。タスクの数を上回る要求があったときは、未処理の要求は、現在実行中のタスクが完了するまで待機します。デフォルトでは、ネットワーク・データベース・サーバ用には 20、パーソナル・データベース・サーバ用は 10 のタスクがあります。サーバ接続の最大数を上回る数のタスクを設定しても、特に利点はありません。

  詳細については、「-gn サーバ・オプション」194 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 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、メモリが限られている環境でのデータベース・サーバのメモリ使用量を節約できます。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。

  詳細については、「-gss サーバ・オプション」196 ページを参照してください。

- **プロセッサの数** 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。デフォルトでは、マシン上のすべてのプロセッサが使用されます。

  詳細については、「-gt サーバ・オプション」197 ページを参照してください。

- **データベース・サーバ・プロセスに割り当てられるスレッドの数** Windows NT/2000/XP では、-gx オプションによって、ファイバをホストして要求を処理するスレッドの数が制御されます。デフォルトでは、これはマシン上の CPU 数よりも 1 つ大きい値に設定されます。各タスクが独自のスレッドである UNIX、NetWare、Windows 95/98/Me では、このオプションは必要ありません。

  詳細については、「-gx サーバ・オプション」198 ページを参照してください。

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。Adaptive Server Anywhere は、ネットワーク通信用と同一マシン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」114 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 (*dbeng9.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、同一マシン通信で使用され、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マシン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 Adaptive Server Anywhere」133 ページ を参照してください。
- **名前付きパイプ**　このプロトコルは、Windows NT でのみ提供されています。名前付きパイプは、同一マシン上にある複数のアプリケーションを、認証基準を満たしたセキュリティ環境で実行させるためのプロトコルです。

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 (*dbsrv9.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、同一マシン通信で使用され、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。
- **名前付きパイプ**　このプロトコルは、Windows NT でのみサポートされます。名前付きパイプは、同一マシン上にある複数のアプリケーションを、認証基準を満たしたセキュリティ環境で実行させるためのプロトコルです。

プロトコルの指定

-x オプションを使用すると、起動時に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サーバに指示できます。次のコマンドは、TCP/IP プロトコルと SPX プロトコルを使用する asademo データベースを起動します。

```
dbsrv9 -x "tcpip,spx" path¥asademo.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。

```
dbsrv9 -x
"tcpip{MyIP=192.75.209.12:2367,192.75.209.32}"
path¥asademo.db
```

-x オプションの一部として使用できるプロトコル・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」276 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。

- データベース・サーバのウィンドウで [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、別のマシン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。

  dbstop オプションの詳細については、「dbstop コマンド・ライン・ユーティリティを使用したデータベース・サーバの停止」 749 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 または NetWare マシンのサーバ表示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、Adaptive Server 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。

   ```
   dbsrv9 -n Ottawa asademo.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。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、そのマシンを使用していれば誰でもサーバのウィンドウで [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」483 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・マシン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。

```
dbeng9 asademo sample
```

警告
データベース・ファイルは、データベース・サーバと同じマシン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。

  詳細については、「組み込みデータベースへの接続」62 ページまたは 「DatabaseFile 接続パラメータ [DBF]」248 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] ー [データベースの開始] を選択する。

  詳細については、『ASA SQL リファレンス・マニュアル』> 「START DATABASE 文 [Interactive SQL]」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。
- -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。

  詳細については、「AutoStop 接続パラメータ [ASTOP]」240 ページ を参照してください。

- Interactive SQL または Embedded SQL で STOP DATABASE 文を使用する。

  詳細については、『ASA SQL リファレンス・マニュアル』>「STOP DATABASE 文」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。これを簡単に行うには、Windows NT/2000/XP または UNIX 用の Adaptive Server Anywhere を、コンピュータからログオフしたときにデータベース・サーバが稼動し続けるように実行します。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。

- **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。

  詳細については、「デーモンとしての UNIX データベース・サーバの稼働」27 ページを参照してください。

- **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。

  詳細については、「Windows サービスの概要」30 ページを参照してください。

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動させるには、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。サーバがハングします。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。

   ```
   dbsrv9 -ud asademo
   ```

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 dbsrv9 asademo
   ```

   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、dbeng9 と dbsrv9 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。

   1 つまたは複数のアプリケーションを実行する前にサーバをデーモンとして起動する必要があるスクリプトまたは自動化プロセスを記述するときは、dbspawn を使用すると便利です。サーバが実行されていることを確認してからアプリケーションを起動できるからです。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 we start any applications
 dbspawn dbsrv9 asademo
 if ( $status != 0 ) then
    echo Failed to start asademo server
    exit
 endif
 # ok, now we 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 we start any applications
 dbspawn dbsrv9 asademo
 if [ $? != 0 ]; then
    echo Failed to start asademo server
    exit
 fi
 # ok, now we 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/sybase/SYBSsa9/bin/dbsrv9",
 "dbsrv9", "-ud", "asademo" );
    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。

詳細については、「-ud サーバ・オプション」215 ページと「プロセス生成ユーティリティ」746 ページを参照してください。

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 NT/2000/XP プログラムと同じように稼動できますが、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 NT/2000/XP 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。Windows NT/2000/XP にログオンできるのは、1 台のコンピュータで一度に 1 人だけです。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 NT/2000/XP 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 Adaptive Server Anywhere サービスを管理します。

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 (*dbsrv9.exe*)
- パーソナル・データベース・サーバ (*dbeng9.exe*)
- SQL Remote Message Agent (*dbremote.exe*)

- Mobile Link 同期サーバ (*dbmlsrv9.exe*)
- サンプル・アプリケーション

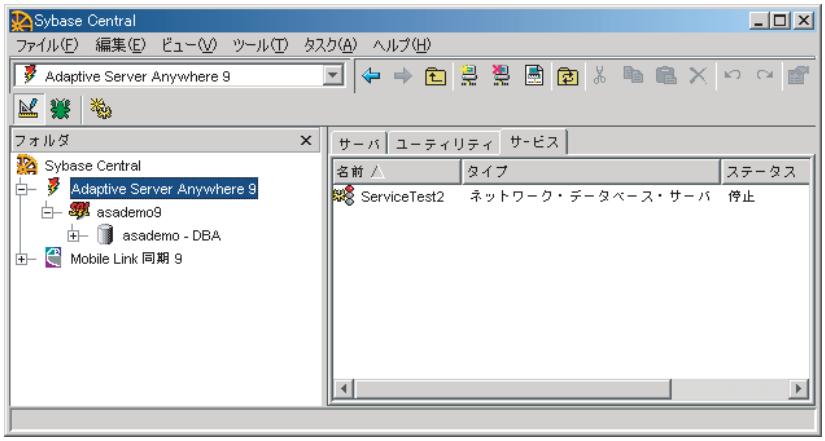
これらのアプリケーションすべてが、Adaptive Server Anywhere のすべての版に付属しているわけではありません。

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、以下のサービス管理タスクを実行できます。

- サービスの追加、編集、削除
- サービスの開始、停止、一時停止
- 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、サーバ・アイコン (実行中、一時停止、停止) によって各サービスの現在の状況を表示します。



サービスの追加

この項では、Sybase Central とサービス作成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。

❖ 新しいサービスを追加するには、次の手順に従います (Sybase Central の場合)。

1 左ウィンドウ枠で、Adaptive Server Anywhere 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] ー [サービス] を選択します。

4 ウィザードの指示に従います。

❖ 新しいデータベースを追加するには、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、サービス作成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・サービスを作成して、指定のパラメータで指定のサーバを起動させるには、次のように入力します。エンジンは LocalSystem ユーザとして実行され、次のコマンドをすべて 1 行で入力します。

```
dbsvc -as -w myserv "C:¥Program Files¥Sybase¥SQL
Anywhere 9¥win32¥dbeng9.exe" -n william -c 8m
"C:¥Program Files¥Sybase¥SQL Anywhere 9¥sample.db"
```

サービス作成ユーティリティとオプションの詳細については、「サービス作成ユーティリティ」738 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。

- このウィンドウで、実行プログラム名自体は付けずに、実行プログラムのオプションを入力します。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 c:¥Program Files¥Sybase¥SQL
  Anywhere 9¥asademo.db
  ```

  改行はオプションです。

  正しいオプションについては、「データベース管理ユーティリティ」641 ページの各プログラムの説明を参照してください。

- 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」37 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。

設定オプションの詳細については、「サービスの設定」34 ページを参照してください。

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。

❖ 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 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。

右ウィンドウ枠にある [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、[ファイル] メニューで [削除] を選択します。

❖ 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
```

サービス作成ユーティリティとオプションの詳細については、「サービス作成ユーティリティ」738 ページを参照してください。

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。

各管理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」641 ページを参照してください。

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. 左ウィンドウ枠で、Adaptive Server Anywhere 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] ー [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、Adaptive Server 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動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 アクセス権を持つユーザが起動したときにのみ起動します。Administrator アクセス権については、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

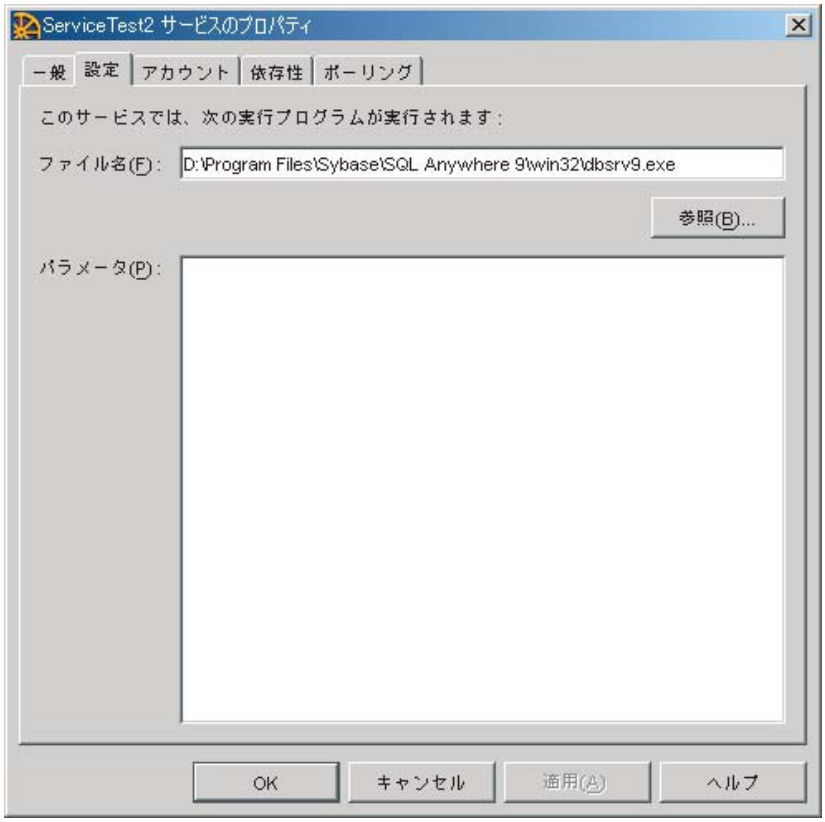
- 2 つのデータベースを実行する my_server というネットワーク・サーバ・サービスを、キャッシュ・サイズが 20 MB で起動するには、[パラメータ] ボックスに次のように入力します。

```
-c 20M
 -n my_server
 c:¥Program Files¥Sybase¥SQL Anywhere 9¥db_1.db
 c:¥Program Files¥Sybase¥SQL Anywhere 9¥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=asademo"
```

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。各プログラムのオプションの詳細については、「データベース・サーバ」153 ページを参照してください。

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することができます。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。これは、Windows のユーザー マネージャから、[高度なユーザー権利] で与えられます。

システム・トレイにアイコンが表示される場合

サービスのアイコンがシステム・トレイとデスクトップのどちらに表示されるかは、選択したアカウントと、[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／オフのどちらに設定したかによって異なります。

- 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することができます。
- 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. 左ウィンドウ枠で、Adaptive Server Anywhere 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. 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 からサーバでデータベースを起動する方法については、『ASA SQL リファレンス・マニュアル』> 「START DATABASE 文 [Interactive SQL]」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、『ASA プログラミング・ガイド』> 「db_start_database 関数」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、停止、一時停止) 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. 左ウィンドウ枠で、Adaptive Server Anywhere プラグインをクリックします。
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. 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、停止、一時停止

❖ サービスを開始、停止、または一時停止するには、次の手順に従います。

1. 左ウィンドウ枠で、Adaptive Server Anywhere 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　サービスを選択し、[ファイル] メニューから [起動]、[停止]、[一時停止] のいずれかを選択します。

一時停止したサービスを再開するには、ポップアップ・メニューから [続行] 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。

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。

サービスを一時停止すると、アプリケーションによる動作が行われなくなります。ただし、アプリケーションは停止せず、データベース・サーバ・サービスではデータベースへのクライアント接続は閉じられません。通常の使用では、サービスを一時停止する必要はありません。

Windows サービス・マネージャ

Adaptive Server 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 NT 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 NT のサービス マネージャで Adaptive Server 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。Adaptive Server 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、Adaptive Server 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、互いに依存し合う複数の実行プログラムをサービスとして実行できます。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	ASANYServer
パーソナル・サーバ	ASANYEngine
SQL Remote Message Agent	ASANYRemote
Mobile Link 同期サーバ	ASANYMobiLink
Replication Agent	ASANYLTM

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することができます。

❖ **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　左ウィンドウ枠で、Adaptive Server Anywhere 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　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] 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。

サービスの依存の管理

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のリストに追加するには、次の手順に従います。

1 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。

2 サービスを右クリックし、ポップアップ・メニューから [プロパティ] を選択します。

3 [依存] タブをクリックします。

4 [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リストに追加します。

5 リストからサービスまたはグループを 1 つ選択します。

6 [OK] をクリックして、そのサービスまたはグループを依存のリストに追加します。

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・サーバの起動時によく発生する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、データベースを新しいバージョンに置き換えると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。この場合、トランザクション・ログとデータベースが食い違い、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

Adaptive Server 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、ASTMP 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。

詳細については、「ASTMP 環境変数」367 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。

問題が発生したり、標準以外のソフトウェアを使用していたり、複数のネットワークを実行している場合は、「クライアント／サーバ通信」113 ページのネットワーク通信の説明をよく読んでください。

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。データベース・サーバ・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。

詳細については、「-z サーバ・オプション」223 ページと 「-o サーバ・オプション」202 ページを参照してください。

正しい asasrv.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、*asasrv.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、Adaptive Server Anywhere は、ユーザが接続文字列に指定した情報を含む新しい *asasrv.ini* ファイルを作成します。*asasrv.ini* ファイルは、Adaptive Server Anywhere の実行プログラム・ディレクトリ (ODBC/DBLib DLL と同じディレクトリ) にあります。

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 *asasrv.ini* がある場合、このファイルをすべて削除します。

- SQL Anywhere のインストール・ディレクトリ (*HKEY_LOCAL_MACHINE¥SOFTWARE¥Sybase¥Adaptive Server Anywhere¥9.0¥Location* レジストリ・キーにリストされたディレクトリで確認可能) の *win32* または *win64* サブディレクトリ
- 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

asasrv.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」94 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。

ログ・ファイルの詳細については、「Logfile 接続パラメータ [LOG]」268 ページを参照してください。

第 2 章

データベースへの接続

この章の内容

この章では、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。ODBC、OLE DB、ADO.NET、iAnywhere JDBC ドライバ、Embedded SQL 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も説明します。また、Sybase Central と Interactive SQL からの接続についても説明します。

Sybase Open Client アプリケーションからのデータベースへの接続については、「Open Server としての Adaptive Server Anywhere」133 ページを参照してください。

JDBC を介した接続 (Sybase Central も Interactive SQL も使用しない場合) の詳細については、『ASA プログラミング・ガイド』> 「JDBC プログラミング」を参照してください。

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 サーバ出力とクライアント LOGFILE 接続パラメータ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。

要求レベル・ログの詳細については、『ASA SQL ユーザーズ・ガイド』>「要求ロギング」を参照してください。

ロックの詳細については、『ASA SQL ユーザーズ・ガイド』>「ロックの仕組み」を参照してください。

接続の確立方法

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 Adaptive Server Anywhere インタフェースのいずれかで関数を呼び出します。Adaptive Server Anywhere が提供するインタフェースを次に示します。

- **ODBC** ODBC 接続については、この章で説明します。
- **ADO.NET** ADO.NET 接続については、この章で説明します。

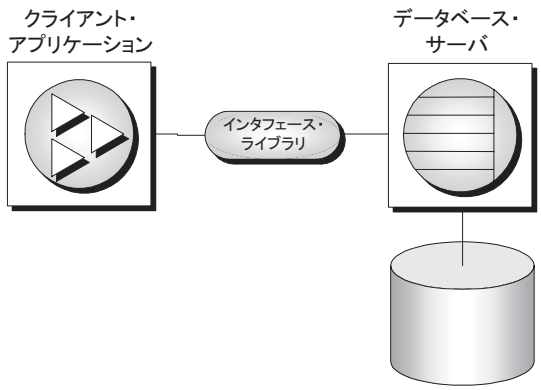
  ADO.NET 接続の詳細については、『ASA プログラミング・ガイド』>「ADO.NET プログラミング・インタフェース」を参照してください。
- **Sybase Open Client** Open Client 接続については、この章では説明しません。

Open Client アプリケーションからの接続については、「Open Server としての Adaptive Server Anywhere」133 ページ を参照してください。

- **jConnect JDBC ドライバ**　Sybase Central と Interactive SQL には、組み込みの接続論理があります。これについては、この章で説明します。jConnect を使用する他のアプリケーションは、この章で説明する接続論理を使用できません。

  JDBC を介した接続については、『ASA プログラミング・ガイド』>「JDBC プログラミング」を参照してください。

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 または Interactive SQL からの接続」55 ページ

目的	参照先 …
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」59 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」70 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」236 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」87 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」113 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」448 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=asademo9;DatabaseName=asademo
```

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)

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。

Adaptive Server 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。照合ユーティリティ (dbcollat) コマンド・ラインの一般的な例を次に示します (すべて 1 行に入力します)。

```
dbcollat -c "uid=DBA;pwd=SQL;dbn=asademo"
c:¥temp¥asademo.col
```

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 Adaptive Server 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。

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」70 ページを参照してください。

Adaptive Server 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。

Interactive SQL、Sybase Central、Adaptive Server Anywhere Console ユーティリティ (dbconsole) では、jConnect を介して接続する場合でも ODBC データ・ソースを使用できます。

Sybase Central または Interactive SQL からの接続

Sybase Central または Interactive SQL を使用してデータベースを管理するには、まず Sybase Central または Interactive SQL に接続します。[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 または Interactive SQL 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、ユーザが自分のマシン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 または Interactive SQL 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、または別のマシン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 または Interactive SQL は、この情報を元に接続するデータベースを識別します。

この項では、Sybase Central と Interactive SQL の [接続] ダイアログにアクセスする方法を説明します。

Sybase Central と Interactive SQL に関する例も含めた接続例の詳細については、「簡単な接続の例」59 ページを参照してください。

[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表示する必要があります。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)。**

- Sybase Central から、[ツール] ー [接続] を選択します。

  Sybase Central プラグインを複数インストールしている場合は、リストから [Adaptive Server Anywhere 9] を選択します。

  または、メイン・ツールバーの [接続] ボタンをクリックするか、[F11] を押して [接続] ダイアログを開きます。

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)。**

- Interactive SQL から、[SQL] ー [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、Adaptive Server Anywhere のサンプル・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 [ASA 9.0 Sample] を選択し、[OK] をクリックします。

接続のドライバの指定

データベースを使って作業しているときは、ユーザからの要求とコマンドはドライバを介してそのデータベースに届きます。Sybase Central と Interactive SQL は、jConnect ドライバと iAnywhere JDBC ドライバという、2 つの主要な JDBC ドライバをサポートしています。両方とも、Adaptive Server Anywhere に付属しています。

iAnywhere JDBC ドライバ

iAnywhere JDBC ドライバは、多くの場合に推奨されるドライバです。これは、Interactive SQL と Sybase Central のデフォルト・ドライバです。Sybase jConnect は、プラットフォームに依存しない JDBC ドライバです。

[接続] ダイアログからデータベースに接続するときに、接続に使用するドライバを [詳細] タブで選択できます。この選択は必須ではありません。iAnywhere JDBC ドライバは、推奨されているドライバであるため、特に選択しないかぎり、すべての接続に自動的に使用されます。

JDBC ドライバの詳細については、『ASA プログラミング・ガイド』>「JDBC ドライバの選択」、『ASA プログラミング・ガイド』>「jConnect JDBC ドライバの使用」、および「ODBC データ・ソースの使用」70 ページを参照してください。

データ・ソースと jConnect ドライバ

原則として、jConnect ドライバは ODBC データ・ソースを使用できません。しかし、Sybase Central と Interactive SQL は例外です。このいずれかで jConnect ドライバを使う場合は、ODBC データ・ソースを指定して接続を確立できます。たとえば、jConnect ドライバを使っていても、ASA 9.0 サンプル・データ・ソースを使用してサンプル・データベースに接続できます。

カスタマイズされたこの機能は、Sybase Central または Interactive SQL で作業しているときのみ使用できます。jConnect を使用して JDBC アプリケーションを構築している場合は、データベースへの接続に ODBC データ・ソースを使用しないでください。

❖ 接続用のドライバを指定するには、次の手順に従います (Sybase Central の場合)。

1. Sybase Central から、[ツール] ー [接続] を選択し、[接続] ダイアログを開きます。
2. ダイアログの [ID] タブと [データベース] タブで、必要な設定を行います。
3. ダイアログの [詳細] タブで、[jConnect 5] または [iAnywhere JDBC ドライバ] を選択します。

[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

Adaptive Server 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 Adaptive Server 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。

使用できる接続パラメータとその使用方法の詳細については、「接続パラメータ」236 ページを参照してください。

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. Sybase Central を起動します。それには、[スタート] メニューから、[プログラム] ー [SQL Anywhere 9] ー [Sybase Central] を選択します。
2. [接続] ダイアログを開きます。それには、[ツール] メニューから [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. [ASA 9.0 Sample] を選択し、[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 Interactive SQL を起動します。それには、[スタート] メニューから、[プログラム] － [SQL Anywhere 9] － [Adaptive Server Anywhere] － [Interactive SQL] の順に選択します。

2 [接続] ダイアログを開きます。それには、[SQL] メニューから [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 [ASA 9.0 Sample] を選択し、[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。

Sybase Central または Interactive SQL から同じマシン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マシンにあるときです。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。
- 使用しているマシン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。
   - サーバにデータベースが 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。
   - サーバに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。

jConnect JDBC ドライバとともにデータ・ソースを使用する方法については、「接続のドライバの指定」56 ページを参照してください。

参照

- 「簡単な接続の例」59 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マシン上で稼働し、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=path¥asademo.db
 uid=DBA
 pwd=SQL
```

path は、Adaptive Server Anywhere インストール・ディレクトリの名前です。

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=dbeng9 -c 8M
 dbf=path¥asademo.db
 uid=DBA
 pwd=SQL
```

参照

- 「[接続] ダイアログを開く」55 ページ
- 「簡単な接続の例」59 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 Adaptive Server Anywhere インタフェースは、データ・ソースを使用できます。ただし、Sybase Central、Interactive SQL、Adaptive Server Anywhere Console ユーティリティ (dbconsole) では、jConnect を使って接続する際にデータ・ソースを使用できるという例外があります。

詳細については、「接続のドライバの指定」56 ページを参照してください。

❖ 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。

ASA 9.0 サンプル・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

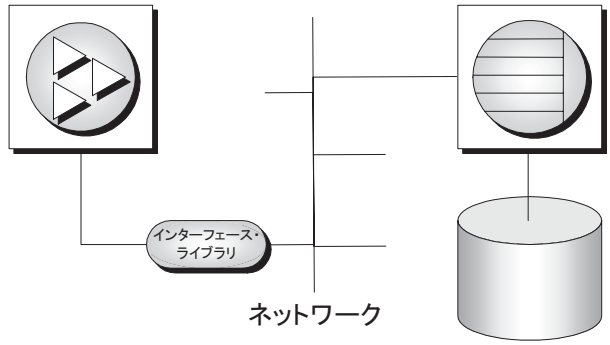
- 「[接続] ダイアログを開く」55 ページ
- 「簡単な接続の例」59 ページ

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。Adaptive Server 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。

ネットワークを介したクライアント／サーバ通信の詳細については、「クライアント／サーバ通信」113 ページを参照してください。



サーバの指定

Adaptive Server 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。

詳細については、「CommLinks 接続パラメータ [LINKS]」243 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 (*asasrv.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、Adaptive Server Anywhere を支援できる接続パラメータがたくさんあります。

詳細については、「ネットワーク・プロトコル・オプション」276 ページを参照してください。

詳細については、「ファイアウォール経由の接続」116 ページを参照してください。

❖ 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] タブで、[サーバ名] 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。

jConnect JDBC ドライバとともにデータ・ソースを使用する方法については、「接続のドライバの指定」56 ページを参照してください。

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

- 「[接続] ダイアログを開く」55 ページ
- 「簡単な接続の例」59 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、マシンに他の Adaptive Server 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
```

デフォルト動作の詳細については、「接続のトラブルシューティング」87 ページを参照してください。

デフォルトを使用しない場合

異なるマシン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 Adaptive Server Anywhere Console 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。

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」87 ページを参照してください。

Adaptive Server 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 Adaptive Server 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。これらのユーティリティは、データベースに接続する場合、「接続のトラブルシューティング」87 ページで示されている手順に従います。

データベース・ツール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・ツール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」641 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。Adaptive Server 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 環境変数の詳細については、「環境変数」363 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 95/98/Me と Windows NT/2000/XP の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。

データ・ソースの格納場所

ODBC 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。Adaptive Server 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
  ```

Adaptive Server Anywhere では、ODBC データ・ソースを使用すると、ODBC インタフェースを使用する Windows アプリケーションの機能も実現できます。

- UNIX 上の Adaptive Server 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 Adaptive Server Anywhere クライアント・インタフェースで使用できます。
- Interactive SQL、Sybase Central、Adaptive Server Anywhere Console ユーティリティ (dbconsole) では、jConnect を使う場合でも ODBC データ・ソースを使用できます。

ODBC データ・ソースの作成

ODBC データ・ソースを Windows 95/98/Me と Windows NT/2000/XP 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。

Adaptive Server 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。

始める前に

この項では、ODBC データ・ソースの作成方法について説明します。データ・ソースを作成する前に、そこに含める接続パラメータを決める必要があります。

詳細については、「簡単な接続の例」59 ページを参照してください。

ODBC アドミニストレータ

Windows 95/98/Me と Windows NT/2000/XP 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. ODBC アドミニストレータを起動します。それには、Sybase Central から [ツール] ー [Adaptive Server Anywhere 9] ー [ODBC アドミニストレータを開く] を選択します。

   別の方法として、[スタート] メニューから、[プログラム] ー [SQL Anywhere 9] ー [Adaptive Server Anywhere] ー [ODBC アドミニストレータ] の順に選択します。

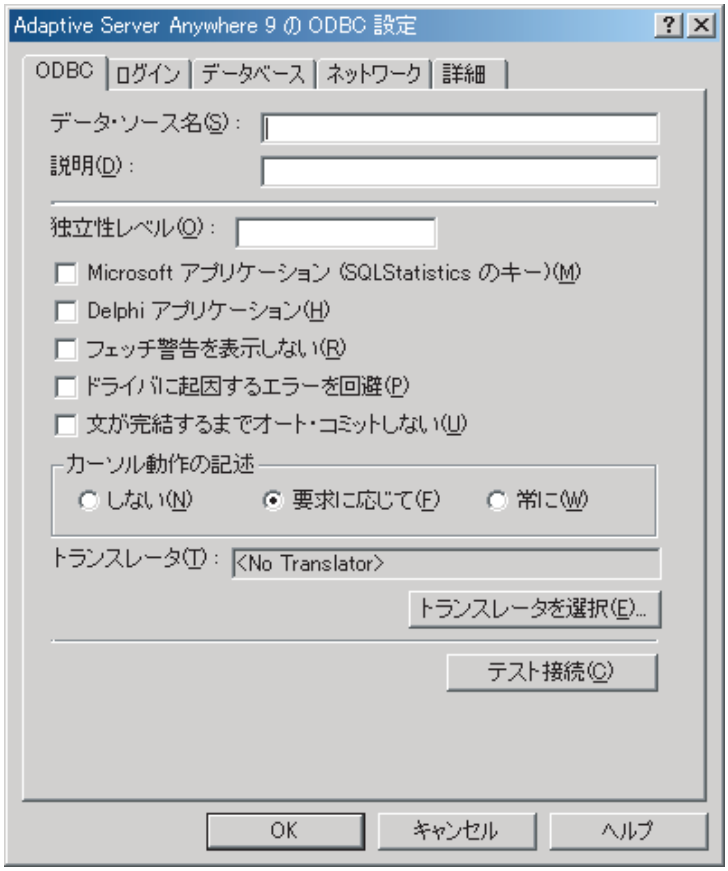
[ODBC データ ソース アドミニストレータ] ダイアログが表示されます。

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 [Adaptive Server Anywhere 9.0] を選択し、[完了] をクリックします。

[Adaptive Server Anywhere 9 の ODBC 設定] ダイアログが表示されます。



このウィンドウのフィールドの多くは、オプションです。各タブのフィールドに関する詳細を表示する場合は、[ヘルプ] ボタンをクリックします。

ダイアログ内のフィールドの詳細については、『SQL Anywhere Studio ヘルプ』> 「[ODBC 設定] ダイアログのヘルプ」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、ウィンドウを閉じてデータ・ソースを作成します。

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。しかし、このユーティリティでは、ファイル・データ・ソースやシステム・データ・ソースを作成することはできません。この 2 つのデータ・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 Adaptive Server Anywhere 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=c:¥Program
Files¥Sybase¥SQL Anywhere 9¥asademo.db"
```

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」662 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) の両方を使用することはできません。

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 [Adaptive Server Anywhere 9.0] 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。

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、*.odbc.ini* ファイルに保管されます。サンプル・ファイルは次のようになります。

```
[My Data Source]
 ENG=myserver
 CommLinks=tcpip(Host=hostname)
 uid=DBA
 pwd=SQL
```

.odbc.ini ファイルには、任意の接続パラメータを入力できます。

接続パラメータの完全なリストについては、「接続パラメータ」236 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・オプションの完全なリストについては、「ネットワーク・プロトコル・オプション」276 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。

詳細については、「ODBC データ・ソースの作成」71 ページと「データ・ソース・ユーティリティ」662 ページを参照してください。

警告

Adaptive Server 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に単純暗号化を追加しないようにしてください。その他のデータ・ソース (Mobile Link 同期など) を使用する場合は、*.odbc.ini* ファイルの内容を読みにくくすると、その他のドライバが正しく機能しなくなる場合があります。

ファイルのロケーション

データベース・サーバは、*.odbc.ini* ファイルを次のロケーションで探します。

1. ODBCINI 環境変数
2. ODBCHOME 環境変数と HOME 環境変数
3. ユーザのホーム・ディレクトリ

4. パス

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、Adaptive Server 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・ファイルを次のロケーションで検索します。

1. ODBC ドライバ (*dbodbc9.dll*) をロードしたディレクトリ。これは、通常 Windows ディレクトリです。

2. レジストリの Adaptive Server Anywhere セクションの Location キーに指定されているディレクトリ。これは、通常 Adaptive Server Anywhere のインストール・ディレクトリと同じです。デフォルトのインストール・ディレクトリは、次のロケーションです。

```
¥Program Files¥Sybase¥ASA
```

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。

ファイル・データ・ソースの詳細については、「Windows でのファイル・データ・ソースの使用」74 ページを参照してください。

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・マシンと 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。

デスクトップ・マシンと Windows CE デバイスの間で Ethernet 接続を使用する場合は、この後説明する手順が有効です。ActiveSync で USB 接続を使用する場合、デスクトップから CE デバイスへの接続はサポートされません。

シリアル・ケーブル接続と ActiveSync 3.x の使用については、「ActiveSync 3.0 とシリアル・ケーブルの使用」78 ページを参照してください。

❖ デスクトップ・アプリケーションから Windows CE データベース・サーバに接続するには、次の手順に従います。

1. サーバの IP アドレスを決定します。-z オプション (出力追加デバッグ情報) を使用して、Windows CE デバイス上でサーバを起動します。

   次に例を示します。

   ```
   dbsrv9 -z -x tcpip -n TestServer asademo.db
   ```

   -z オプションを使用して、サーバは起動中に IP アドレスを書き出します。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。

   CE デバイスの IP 割り当てを静的か動的のどちらかに変更するには、Windows の [コントロール パネル] で設定します。[ネットワーク] を開き、[サービス] タブを選択します。[リモート・アクセス・サービス] を選択し、[プロパティ] ー [ネットワーク] ー [TCP/IP 設定] をクリックします。

2 使用しているデスクトップ・マシンで ODBC プロファイルを作成します。

ODBC アドミニストレータを開き、[追加] をクリックします。ドライバ・リストから [Adaptive Server Anywhere 9.0] を選択し、[完了] を選択します。[Adaptive Server Anywhere 9 の ODBC 設定] ダイアログが表示されます。

- [ログイン] タブで、ユーザ ID とパスワードを入力します。
- [データベース] タブで、サーバ名を入力します。
- [ネットワーク] タブで、[TCP/IP] チェックボックスをオンにし、隣接するフィールドに次の文字列を入力します。

```
dobroadcast=DIRECT;host=XXX.XXX.XXX.XXX
```

XXX.XXX.XXX.XXX には、サーバの IP アドレスを入力します。

- [ODBC] タブで、[テスト接続] をクリックして ODBC データ・ソースが適切に設定されていることを確認します。

3 ODBC アドミニストレータを終了します。

4 使用している Windows CE マシンでデータベース・サーバが実行されているのを確認します。

5 使用しているデスクトップ・マシンで、Interactive SQL などのアプリケーションを起動し、作成した ODBC データ・ソースを選択します。アプリケーションが Windows CE データベースに接続します。

ActiveSync 3.0 とシリアル・ケーブルの使用

シリアル・ケーブルを介してデスクトップから Windows CE デバイスに接続するには、Adaptive Server Anywhere で TCP/IP プロトコルを使用します。このコンテキストで TCP/IP を使用するには、ActiveSync

のインストール環境が、デスクトップ・マシンと Windows CE デバイスの間でリモート・アクセス・サービス (RAS) リンクを提供するように設定されている必要があります。

詳細については、「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」77 ページを参照してください。

ActiveSync 3.x では、RAS のインストールと設定は行われません。シリアル接続を介した TCP/IP 接続を実行するには、RAS を自分でインストールする必要があります。

RAS をインストールするための指示は、Microsoft によって提供されています。http://support.microsoft.com/support/kb/articles/Q241/2/16.ASP (Microsoft Knowledge Base、記事 Q241216) から入手できます。Windows NT サービス・パックの再インストールやユーザ・マネージャを使用したユーザ・アカウントのダイヤルイン・アクセスの保証を含め、指示を*正確に*実行してください。

指示に従っていくと、モデムのインストールを求めるメッセージが表示されます。このとき、[2 台の PC 間のダイヤルアップ・ネットワーキング・シリアル・ケーブル] を選択してください。

ActiveSync での RAS の使用

次のリストには、RAS を使用したシリアル接続で ActiveSync 3.0 接続を有効にするための提案が示されています。

1. ActiveSync の [接続設定] では、[デスクトップ・コンピュータとのネットワーク (Ethernet) とリモート・アクセス・サービス (RAS) サーバ接続を可能にする] チェックボックスをオンにします。[この COM ポートへのシリアル・ケーブル接続または赤外線接続を可能にする] チェックボックスはオフにする必要があります。

2. デスクトップで、(Windows NT の管理ツールの下にある) リモート・アクセス・アドミニストレータを使用して、COM1 で RAS を起動します。

3. Windows CE デバイスで、ActiveSync クライアント (Windows CE PC 上の *repllog.exe*) を実行します。[シリアル接続] を選択します。

4. 接続が確立されるまで待機します。

5. Windows NT で *ipconfig* ユーティリティをテスト実行し、デバイスの静的 IP (192.168.55.100) を確認します。これは、たとえば CE デバイス上で実行中の Adaptive Server Anywhere データベース・サーバに接続するときに使用する IP です。

6. デバイスを切り替える場合は、RAS サービスを停止して再起動します (またはリブートします)。

7. 上記のとおりにすべての設定を行ったにもかかわらず、デバイスからデスクトップへの接続が失敗する場合は、ポート設定が [コントロール パネル] の [モデム] アプレットのボー・レートと一致しているかどうかを確認してください。

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 Adaptive Server Anywhere 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 Adaptive Server 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、Adaptive Server Anywhere を PowerBuilder OLE DB データ・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。

OLE DB の詳細については、『ASA プログラミング・ガイド』> 「OLE DB の概要」を参照してください。

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。Adaptive Server 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 ASA OLE DB プロバイダ** Adaptive Server Anywhere OLE DB プロバイダは、OLE DB データ・ソースとしての Adaptive Server Anywhere へのアクセスを提供します。このとき、ODBC コンポーネントを必要としません。このプロバイダの省略名は、**ASAProv** です。

  ASAProv 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 **ASAProv** 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。

  OLE DB プロバイダの詳細については、『ASA プログラミング・ガイド』> 「OLE DB の概要」を参照してください。

- **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 Adaptive Server Anywhere ODBC ドライバが必要です。

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 Adaptive Server 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。

ADO の詳細については、『ASA プログラミング・ガイド』>「Adaptive Server Anywhere を使用した ADO プログラミング」を参照してください。

例

次の Visual Basic コードは、Adaptive Server 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 = "ASAProv"
 myConn.ConnectionString = "Data Source=ASA 9.0 Sample"
 myConn.Open
```

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。

- **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、Adaptive Server Anywhere の Java 機能を使用している場合は、StartLine (START) 接続パラメータに追加のキャッシュ・メモリを指定してください。次に示す組み込みデータベース接続パラメータの例は、Java 機能を使用する接続を示しています。

```
DBF=path¥asademo.db
 dbn=Sample
 ENG=Sample Server
 uid=DBA
 pwd=SQL
 Start=dbeng9 -c 8M
```

接続パラメータのリストについては、「接続パラメータとネットワーク・プロトコル・オプション」235 ページを参照してください。

接続文字列の文字セットに関する問題については、「接続文字列と文字セット」448 ページを参照してください。

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 のいずれかであり、false の場合は NO、OFF、0、FALSE のいずれかです。
- **大文字と小文字の区別**　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 ～ 127 の範囲の ASCII 文字セットで構成します。他のパラメータには、このような制限はありません。

  文字セットの問題の詳細については、「接続文字列と文字セット」448 ページを参照してください。
- **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 と OLE DB はこの例外です。すなわち、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、Adaptive Server 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、Adaptive Server 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」113 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:odbc:** 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 Adaptive Server Anywhere データ・ソースを指定するか、**Driver=Adaptive Server Anywhere 9.0** (UNIX と Linux では、このパラメータは **Driver=libdbodb9** として指定) を含めてください。

トラブルシューティングのヒントについては、「サーバ起動時のトラブルシューティング」45 ページと「ネットワーク通信のトラブルシューティング」129 ページを参照してください。

接続を確立する手順

接続を確立するため、Adaptive Server 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、Adaptive Server 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** Adaptive Server Anywhere は、接続パラメータを使用して、マシン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** Adaptive Server 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** Adaptive Server 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。

次の項では、これらの各手順を詳細に説明します。

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、Adaptive Server 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。ここでは、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 Adaptive Server 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、*.odbc.ini* ファイルまたはレジストリで指定されたデータ・ソースを探します。ODBC アドミニストレータを使用してデータ・ソースを作成すると、Adaptive Server Anywhere は ODBC ドライバに現在のロケーションを入力します。

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。Adaptive Server 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 NT/2000/XP と Windows 95/98/Me**　*dblib9.dll*
- **UNIX**　オペレーティング・システム固有の拡張子を持つ *dblib9*
- **NetWare**　*dblib9.nlm*

OLE DB ドライバのロケーション

ASAOLEDB 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 (ASAProv) が使用されます。エントリは、ASAProv のインストール時、または登録時に作成されます。

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をシステム・パス内に含めます。

- **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。

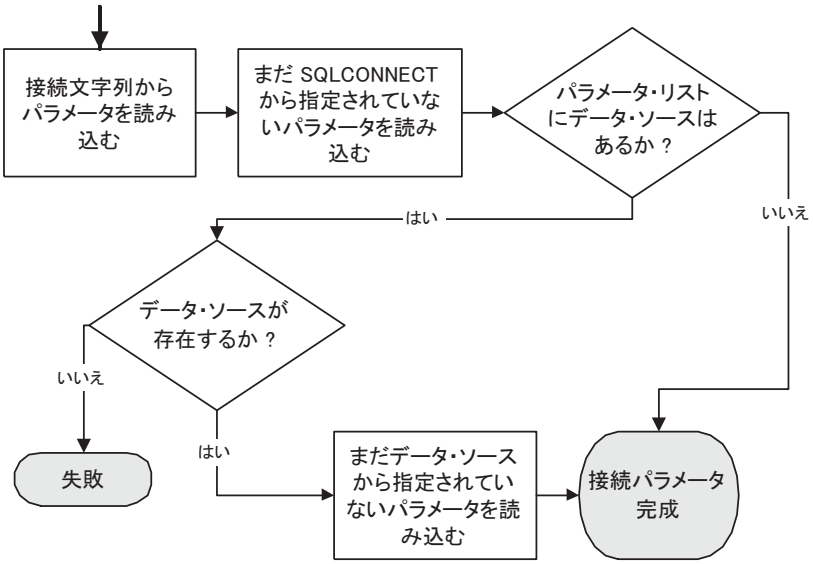
- **UNIX オペレーティング・システム** UNIX では、システム・パスとユーザ・ライブラリ・パスからファイルを探します。
- **NetWare** NetWare では、検索パスと *sys:system* ディレクトリからファイルを探します。

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

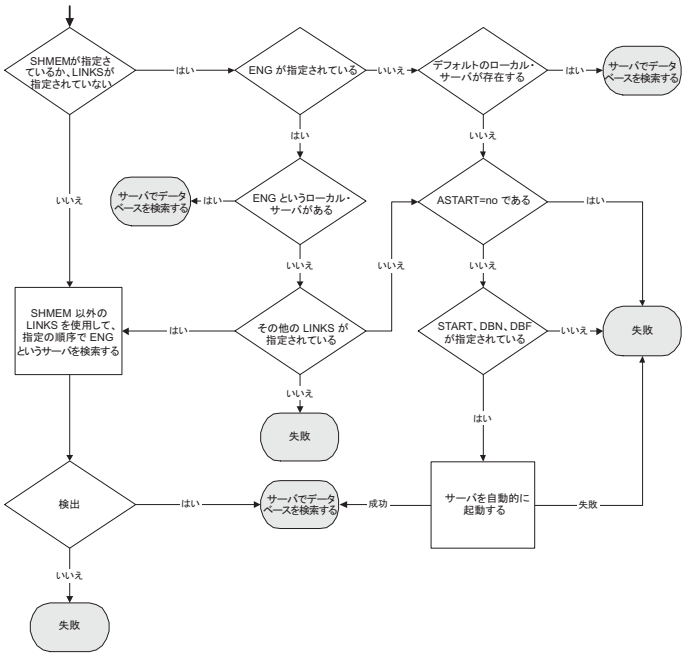
つまり、パラメータがデータ・ソースと接続文字列の両方に指定された場合、接続文字列の値がデータ・ソースの値よりも優先されます。

- **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。

サーバの検出

接続確立処理の次の段階では、Adaptive Server 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、Adaptive Server Anywhere はデフォルト・サーバを探します。



Adaptive Server Anywhere はサーバを検出すると、要求された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。

詳細については、「データベースの検出」93 ページを参照してください。

Adaptive Server 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。

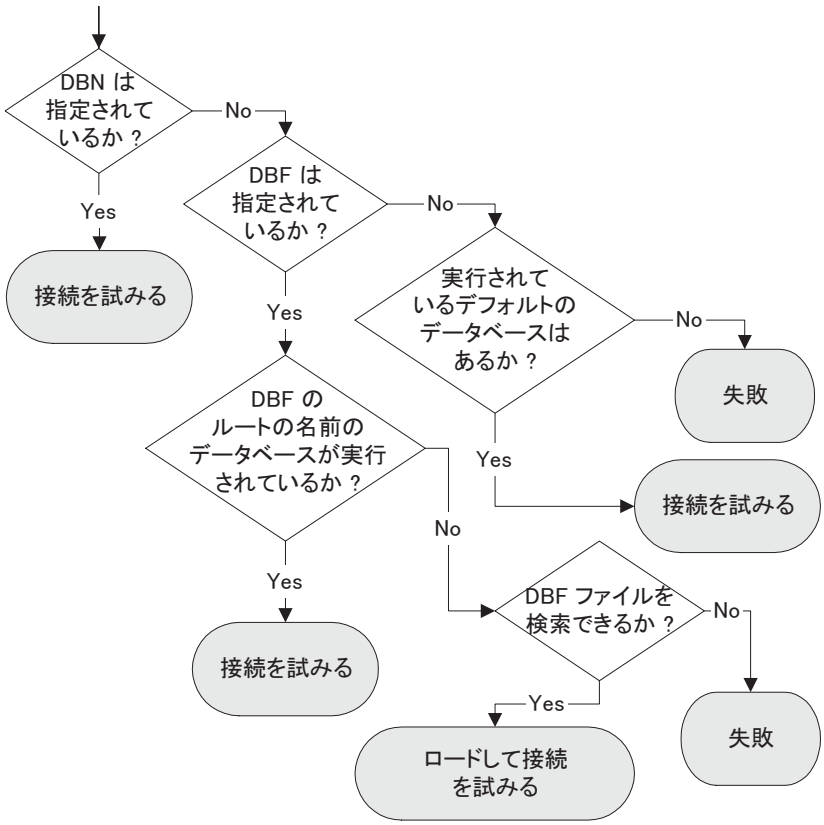
注意

- 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。
- ネットワークの検索では、Adaptive Server Anywhere がサポートする 1 つまたは複数のプロトコルも検索されます。各プロトコルに対して、ネットワーク・ライブラリは単一のポートを起動します。そのプロトコルを介する接続はすべて、常に単一ポートを使用します。

- CommLinks (LINKS) 接続パラメータへの引数には、各ネットワーク・ポート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。Adaptive Server 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。

データベースの検出

Adaptive Server 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) 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・オプションの詳細については、「ネットワーク・プロトコル・オプション」276 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 *asasrv.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]
 Link=protocol_name
 Address=address_string
```

asasrv.ini ファイルは、Adaptive Server Anywhere の実行プログラム・ディレクトリ (ODBC/ DBLib DLL と同じディレクトリ) にあります。

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 *asasrv.ini* ファイルの内容を読みにくくすることができます。

詳細については、「.ini ファイルの内容の非表示」679 ページを参照してください。

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、Adaptive Server 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 ユーティリティの動作は 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。

詳細については、『ASA SQL リファレンス・マニュアル』>「CONNECT 文 [ESQL] [Interactive SQL]」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・サーバが現在のマシン上で使用できるかどうかをテストします。

```
dbping
```

dbping オプションの詳細については、「Ping ユーティリティ」730 ページを参照してください。

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。

サポートされているオペレーティング・システム

統合化ログイン機能は、Windows NT/2000/XP サーバでのみ利用できます。Windows 95/98/Me クライアントや Windows NT/2000/XP クライアントは、統合化ログインを使用して Windows NT/2000/XP で動作するネットワーク・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、マシン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、マシン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することができます。次のような利点があります。

- 統合化ログインを使用してデータベースに接続するときは、ユーザ ID またはパスワードを入力する必要がありません。
- 統合化ログインを使用する場合、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、マシン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 にマッピングすることができます。

- Windows NT/2000/XP マシン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。

セキュリティおよび統合化ログインの詳細については、「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」107 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。

ログイン時、指定された Windows ユーザが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・マッピングのある Windows ユーザ・グループに所属する場合、そのユーザは、データベース・ユーザ、または 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・グループのメンバである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方法については、「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、マシン上の複数の 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、マシン上の 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 またはグループ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 またはグループ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 ntgroupA と ntgroupB に属する Windows ユーザ windowuser を取り上げます。

SQL 文	許可内容
`GRANT INTEGRATED LOGIN windowsuser AS USER dbuserA`	windowuser 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 ntgroupA AS USER dbuserB`	windowsuser は、ntgroupA 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 ntgroupA AS USER dbuserB GRANT INTEGRATED LOGIN ntgroupb 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 ntgroupA AS USER dbuserA GRANT INTEGRATED LOGIN ntgroupb 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・グループのメンバーシップを確認するために、Adaptive Server Anywhere データベース・サーバを実行中のマシンが使用されます。Domain Controller サーバが、データベース・サーバを実行中のマシンとは異なる場合は、[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'
```

詳細については、「INTEGRATED_SERVER_NAME オプション [データベース]」853 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。

1. 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。

2. 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することができます。

パスワードなしでユーザ ID への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あるだけでなく、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。パスワードのないデータベース・ユーザ ID は、データベースに接続できません。

❖ パスワードなしでユーザ ID への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 パスワードなしでデータベース・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 Mixed または Integrated (大文字と小文字の区別はなし) のどちらかに設定することで、データベース内で統合化ログイン機能を使用できるようにします。この手順には、DBA 権限が必要です。

2　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 [統合化ログイン] フォルダに表示されます。

3　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・オプションは、次の 3 つの値のどれかを受け入れます。大文字と小文字は区別しません。

- **Standard**　統合化ログインは許可されません。これはデフォルト設定です。統合化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。
- **Mixed**　この設定では、統合化ログインと標準ログインの両方を使用できます。
- **Integrated**　データベースへのすべてのログインを、統合化ログインを使用して行います。

警告

LOGIN_MODE データベース・オプションを Integrated に設定すると、接続できるのは、統合化ログイン・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。

例

次の SQL 文は、LOGIN_MODE データベース・オプションの値に Mixed を設定し、標準ログインと統合化ログインの両方の接続を許可します。

```
SET OPTION Public.LOGIN_MODE = Mixed
```

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 権限を持つ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 [統合化ログイン] フォルダを開きます。
3. [ファイル] メニューから [新規] ー [統合化ログイン] を選択します。
4. ウィザードの最初のページに、統合化ログインが作成されるシステム (コンピュータ) のユーザ名またはグループ名を入力します。

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 権限を使用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 NT 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
```

詳細については、『ASA SQL リファレンス・マニュアル』> 「GRANT 文」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 mywindowsusers のメンバである Window NT ユーザは、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 mywindowsusers
 AS USER DBA
```

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 権限を使用してデータベースに接続します。

2 [統合化ログイン] フォルダを開きます。

3 右ウィンドウ枠で、統合化ログイン・パーミッションを削除するユーザまたはグループを右クリックし、ポップアップ・メニューから [削除] を選択します。

❖ 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 権限を使用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 NT 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
```

詳細については、『ASA SQL リファレンス・マニュアル』>「REVOKE 文」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 Mixed 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。

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、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、統合化ログイン接続を受け入れることができる。
- 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 ユーティリティ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。

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、Adaptive Server Anywhere セキュリティ・システムではなく、Windows NT/2000/XP のログイン制御システムを使用して動作します。基本的に、データベースをホストしているマシンにログインでき、「統合化ログインの使用方法」97 ページに説明されているその他の条件を満たしていれば、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 NT/2000/XP 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 NT/2000/XP 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。

特に、Windows NT のワークステーションやサーバがインストールされている場合、デフォルトで "Guest" ユーザ・プロファイルが作成されて、有効になることに注意してください。

警告
ユーザ・プロファイル Guest を有効にしておくと、そのサーバがホストになってい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、データベースをホストしている Windows NT 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 NT 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、Adaptive Server Anywhere データベースのホストとなるすべての Windows NT/2000/XP マシンで、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。

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

次の SQL 文を使用して、データベースの LOGIN_MODE オプションの値を Mixed または Integrated に設定すると、そのデータベースに対する統合化ログインが永続的に有効になります。

```
SET OPTION Public.LOGIN_MODE = Mixed
```

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 オプションを一時的に変更しても、ユーザは統合化ログインを介してアクセスできます。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 = Mixed
```

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マシン (ユーザのマシンなど) にコピーされると、

データベースへのアクセスには Adaptive Server 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マシン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。

SET OPTION 文の使用方法の詳細については、『ASA SQL リファレンス・マニュアル』> 「SET OPTION 文」を参照してください。

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・マシンとサーバの両方に存在する必要があります。両方のマシンで、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・マシンとそのデータベースをホストしているアプリケーション・サーバ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。jsmith は、ローカル・マシンとネットワーク・サーバをホストしているサーバの両方にログインが許可されている必要があります。

- 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・マシンのユーザ・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。

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。

第 3 章

クライアント／サーバ通信

この章の内容

各ネットワーク環境にはそれぞれの特性があります。この章では、ネットワーク通信のデータベース・サーバの機能の適切な使用に関する面と、ネットワーク通信問題を診断するためのいくつかのヒントを説明します。また、ネットワークを操作する方法と、各プロトコル上でネットワーク・データベース・サーバを実行するためのヒントを説明します。

ネットワーク・データベース・サーバのみ
この章の内容は、ネットワーク・サーバにのみ適用されます。パーソナル・データベース・サーバを使用している場合は、この章を読む必要はありません。

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 Adaptive Server Anywhere 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 NT/2000/XP と Windows 95/98/Me** TCP/IP または SPX 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

さらに、同じマシンで実行されているクライアント・アプリケーションから通信するために、名前付きパイプ・プロトコルが提供されます。

- **Windows NT/2000/XP/Windows Server 2003** 名前付きパイプは、同一マシン上であれば、どのアプリケーションとサーバ間の通信にでも使用できますが、通常はこの目的ではおすすめできません。名前付きパイプはネットワーク通信には使用されませんが、C2 セキュリティ基準を満たした設定と同等の方法で Adaptive Server Anywhere を実行するために使用できます。

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。Adaptive Server 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、元来、カリフォルニア大学バークレー校 (University of California at Berkeley) が BSD UNIX 向けに実装したプロトコル・スイートでした。TCP/IP は、Internet と WWW の普及に伴い広く使用されるようになりました。

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。Adaptive Server Anywhere では、初期サーバ名解析と、接続と通信用の TCP の実行に IP の最上部の UDP 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_TLS (以前の Certicom) または RSA_TLS 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。

詳細については、『SQL Anywhere Studio セキュリティ・ガイド』>「Adaptive Server Anywhere トランスポート・レイヤ・セキュリティ」を参照してください。

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 2.0 を使用します。Windows NT/2000/XP 上のクライアントの TCP/IP 実装でも、Winsock 2.0 を使用します。Windows 95/98/Me 上のクライアントは、デフォルトで Winsock 1.1 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。

詳細については、「DLL プロトコル・オプション」281 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。

詳細については、「-p サーバ・オプション」203 ページまたは「CommBufferSize 接続パラメータ [CBSIZE]」241 ページ を参照してください。

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。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。
- **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。
- **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、Adaptive Server Anywhere サーバのアドレスとすべての Adaptive Server Anywhere クライアントのアドレス間の TCP/IP トラフィックを許可するように設定します。Adaptive Server Anywhere サーバのアドレスは、Adaptive Server Anywhere サーバを実行中のマシンの IP アドレス (HOST パラメータ) と Adaptive Server 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 Adaptive Server Anywhere クライアントのアドレスは、クライアント・マシンの 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・ポートが拒否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・マシン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。

詳細については、「ClientPort プロトコル・オプション [CPORT]」279 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 ～ 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** のマシン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]」243 ページ
- 「ClientPort プロトコル・オプション [CPORT]」279 ページ
- 「ServerPort プロトコル・オプション [PORT]」294 ページ
- 「Host プロトコル・オプション [IP]」284 ページ
- 「DoBroadcast プロトコル・オプション [DOBROAD]」282 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。

  詳細については、「Host プロトコル・オプション [IP]」284 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。

  詳細については、「DoBroadcast プロトコル・オプション [DOBROAD]」282 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。

  詳細については、「MyIP プロトコル・オプション [ME]」292 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。

詳細については、「Timeout プロトコル・オプション [TO]」298 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・セキュリティにより、サーバ認証、ECC_TLS (以前の Certicom) または RSA_TLS 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。

詳細については、『SQL Anywhere Studio セキュリティ・ガイド』> 「Adaptive Server Anywhere トランスポート・レイヤ・セキュリティ」を参照してください。

LDAP サーバを使用した接続

Windows (CE と Wind64 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サーバを追跡することができます。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。IP アドレス (HOST=) を指定する必要もありません。サーバ検索ユーティリティ (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 と共に使用し、ネットワーク・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サーバ・マシンと各クライアント・マシンに作成してください。デフォルトで、このファイル名は *asaldap.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 Adaptive Server Anywhere 実行プログラム (たとえば、Windows の場合 *%asany%¥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=<machine 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>
```

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 *asasrv.ini* ファイルの内容を読みにくくすることができます。

詳細については、「.ini ファイルの内容の非表示」679 ページを参照してください。

server LDAP サーバを実行中のマシンの名前または 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。

basedn Adaptive Server 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・オブジェクトを指定します。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検索 [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 0 はこのオプションを無効にして、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。

例

次に、*asaldap.ini* ファイルの例を示します。

```
[LDAP]
 server=ldapserver
 basedn=dc=iAnywhere,dc=com

authdn=cn=ASAServer,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 Adaptive Server 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
```

サーバが起動するときに、LDAP ファイル内で同じ名前を持つ既存のエントリをチェックします。1 つ見つかると、次のいずれかの場合に置き換えられます。

- LDAP のロケーション・エントリが起動しようとするサーバと一致する。
- LDAP エントリ内のタイムスタンプ・フィールドが 10 分以上古い (タイムアウト値が設定可能)。

このいずれも該当しない場合、起動しようとしている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、サーバが 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。これらの値はいずれも設定可能です。

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、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。

サーバ検索ユーティリティ (dblocate) も LDAP を使用します。LDAP にリストされているすべてのサーバが、返されるサーバのリストに追加されます。これにより、サーバ検索ユーティリティ (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。NetWare、Windows NT/2000/XP と Windows 95/98/Me 用の Adaptive Server Anywhere はすべて、SPX プロトコルを採用できます。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

一部のマシンでは NetWare バインダリを使用できます。これには、Client Service for NetWare がインストールされている NetWare サーバや、Windows NT/2000/XP または Windows 95/98/Me マシンがあります。これらのマシン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マシン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 (IP) 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。

  詳細については、「Host プロトコル・オプション [IP]」284 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。

  詳細については、「DoBroadcast プロトコル・オプション [DOBROAD]」282 ページ を参照してください。

名前付きパイプの使用

名前付きパイプは、プロセス間通信のための機能です。名前付きパイプは、同じコンピュータ上または異なるコンピュータ上のプロセス間の通信に使用できます。

Adaptive Server Anywhere Windows NT/2000/XP 版では、同一マシン通信にはローカルの名前付きパイプを使用できます。

詳細については、「サポートされているネットワーク・プロトコル」 114 ページを参照してください。

共有メモリ (デフォルトのプロトコル) は、名前付きパイプよりも効果的であることに留意してください。名前付きパイプ・プロトコルは、C2 セキュリティ基準を満たした設定と同等の方法で Adaptive Server Anywhere を実行する場合に使用してください。

Adaptive Server Anywhere は、異なるマシン間のクライアント／サーバ通信には、名前付きパイプを使用しません。

パフォーマンス改善のための通信圧縮設定の調整

ある状況では、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、Adaptive Server 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。

特定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をチューニングする高度な方法は、CommBufferSize (CBSIZE) 接続パラメータと CompressionThreshold (COMPTH) 接続パラメータのいずれか、またはその両方を調整することです。

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」914 ページまたは『ASA SQL リファレンス・マニュアル』>「sa_conn_compression_info システム・プロシージャ」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、Adaptive Server 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 Adaptive Server 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、Adaptive Server 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]」245 ページ と 「-pc サーバ・オプション」203 ページを参照してください。

CommBufferSize 設定の変更

CommBufferSize (CBSIZE) 自体は圧縮パラメータではありませんが、調整すると圧縮機能が向上します。特にクライアントとサーバ間で大量のデータを転送する場合に効果を発揮します。パケット・サイズを大きくすると、複数のローのフェッチ、長いローのフェッチ、BLOB の挿入または取り出しのパフォーマンスが向上します。

ただし、いつもそうであるように、パフォーマンスの向上は良い点ばかりではありません。それぞれの接続は、バッファのプールを持っています。バッファ・サイズを大きくすると、メモリの使用率も増大します。特定の状況を分析して、CommBufferSize を増加するコストに見合うだけの利点があることを確認してください。

詳細については、「TCP/IP パフォーマンスのチューニング」116 ページ、「CommBufferSize 接続パラメータ [CBSIZE]」241 ページ、または「-p サーバ・オプション」203 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

Adaptive Server 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、Adaptive Server 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。

詳細については、「CompressionThreshold 接続パラメータ [COMPTH]」247 ページ と 「-pt サーバ・オプション」204 ページを参照してください。

❖ Adaptive Server Anywhere の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]」245 ページ と 「-pc サーバ・オプション」203 ページを参照してください。

2　CommBufferSize 設定を調整します。

Adaptive Server Anywhere のパケット・サイズを大きくすると、圧縮のパフォーマンスを向上させることができます。

CommBufferSize 設定の調整については、「CommBufferSize 接続パラメータ [CBSIZE]」241 ページ または 「-p サーバ・オプション」203 ページを参照してください。

3　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]」247 ページ と 「-pt サーバ・オプション」204 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。

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 オプションを使って、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って、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。

-x オプションの詳細については、「-x サーバ・オプション」218 ページ を参照してください。

CommLinks (LINKS) 接続パラメータの詳細については、「CommLinks 接続パラメータ [LINKS]」243 ページ を参照してください。

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 つの整数をピリオドで区切った数字 (191.72.109.12 など) を使って、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 つのパケットを送ろうとします。

まず、自分に対して ping を行うことによって、自分自身のマシンが正常に設定されているかどうかを調べます。IP アドレスが 191.72.109.12 の場合、次のように入力します。

```
ping 191.72.109.12
```

コマンド・プロンプトで入力して、パケットがルート指定されるかどうかを調べます。パケットが送られると、次のように表示されます。

```
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
 ...
```

これが動作すれば、このコンピュータはパケットを自分自身に送れます。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、そのユーザに ping 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。

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。同じようなマシン上で同じ設定を使って、問題を再現してみます。あるマシン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」87 ページと「サーバ起動時のトラブルシューティング」45 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、クライアントが要求をブロードキャストしてデータベース・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) ネットワーク上のマシン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マシン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。

  詳細については、「ファイアウォール経由の接続」116 ページを参照してください。

- 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。

- ローカル・マシン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・マシン以外のマシン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」276 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。

活性タイムアウト値の詳細については、「LivenessTimeout 接続パラメータ [LTO]」267 ページ と 「-tl サーバ・オプション」211 ページを参照してください。

接続タイムアウトの詳細については、「Idle 接続パラメータ [IDLE]」263 ページ と 「-ti サーバ・オプション」210 ページを参照してください。

第 4 章

Open Server としての Adaptive Server Anywhere

この章の内容

Adaptive Server 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 Adaptive Server Anywhere データベースにネイティブに接続できます。

この章では、Open Server として Adaptive Server Anywhere を使用する方法と、Open Client と Adaptive Server Anywhere が一緒に動作するように設定する方法を説明します。

Open Client アプリケーションを開発して Adaptive Server Anywhere と共に使用する方法については、『ASA プログラミング・ガイド』> 「Open Client インタフェース」を参照してください。

Open Client、Open Server、TDS

この章では、Adaptive Server Anywhere が Sybase Open Client/Open Server のクライアント／サーバ・アーキテクチャにどのように適合するかを説明します。この項でアーキテクチャの主要な概念について説明し、次に、章の残りの部分でその概念のバックグラウンドについて説明します。

単に、Sybase のアプリケーションを Adaptive Server Anywhere と共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

Adaptive Server 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。

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、(Sybase 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 Adaptive Server 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

Adaptive Server 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、Adaptive Server 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。Adaptive Server Anywhere は、TCP/IP ネットワーク・プロトコルを介してのみ TDS をサポートします。それとは対照的に、Adaptive Server Anywhere 固有のアプリケーション・プロトコルは、同一マシン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 Adaptive Server Anywhere

Adaptive Server 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 Adaptive Server Anywhere と共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、Adaptive Server 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 Adaptive Server 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」603 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れば、Adaptive Server Anywhere は OmniConnect 用のデータ・ソースとして動作できます。

Adaptive Server 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 Adaptive Server 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

Adaptive Server 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

Adaptive Server 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。

- **Adaptive Server 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 (*dbsrv9.exe*) を使用します。パーソナル・サーバ (*dbeng9.exe*) を Open Server として使用できるのは、同一のマシン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、Adaptive Server 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 Adaptive Server 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。

- **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 Studio 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

Adaptive Server 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。

```
dbsrv9 -x tcpip,spx asademo.db
 dbsrv9 -x tcpip -n myserver asademo.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、同一マシン上の通信用 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、Adaptive Server 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します。

ポート番号

あるマシン上で 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。Adaptive Server Anywhere のデフォルトのポート番号は 2638 です。Adaptive Server 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。

```
dbsrv9 -x tcpip(ServerPort=2629) -n myserver asademo.db
```

複数のローカル・データベース・サーバが実行中のとき、またはネットワーク・サーバに接続するときは、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・サーバが稼働中のマシン名と、使用する TCP/IP ポートを、クライアント・マシン側の interfaces ファイルに指定します。

クライアント・マシンの設定の詳細については、「Open Server の設定」140 ページを参照してください。

Open Server の設定

Adaptive Server 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、住所録と同じように、マシン上の 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をリストします。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。

Adaptive Server Anywhere をインストールすると、セットアップ・プログラムによって簡単な 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 Adaptive Server 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 Adaptive Server 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 の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。

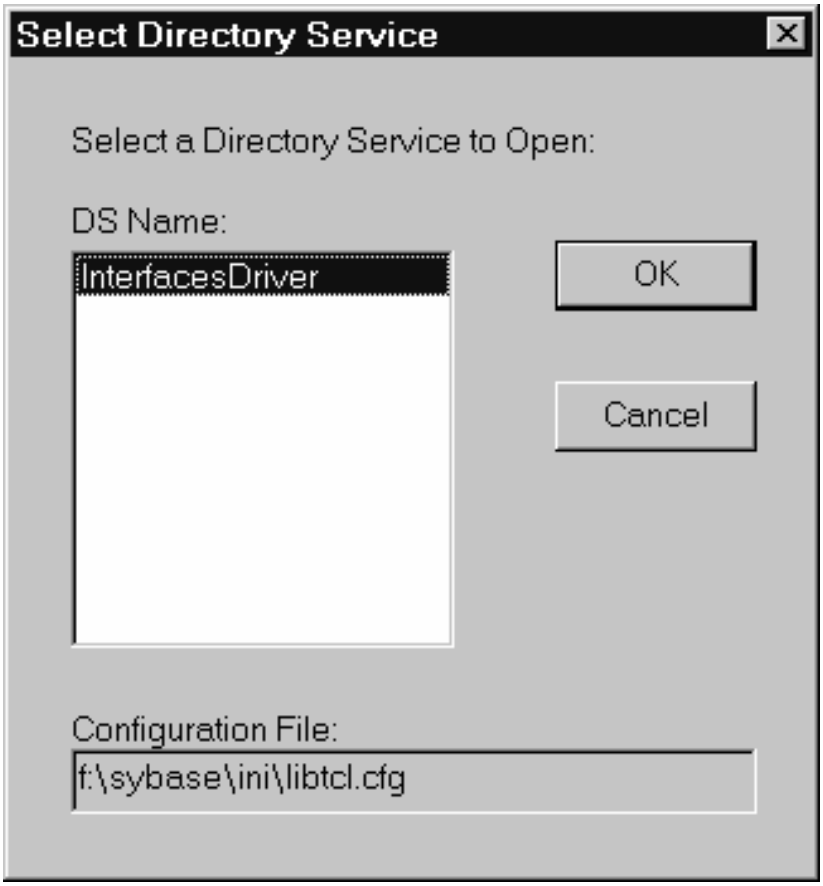
以下の項では、Adaptive Server Anywhere に必要なタスクのために *DSEdit* を使用する方法を説明します。これは、*DSEdit* ユーティリティの完全なマニュアルではありません。

DSEdit の詳細については、他の Sybase 製品に同梱されている、使用プラットホームに対応した『*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

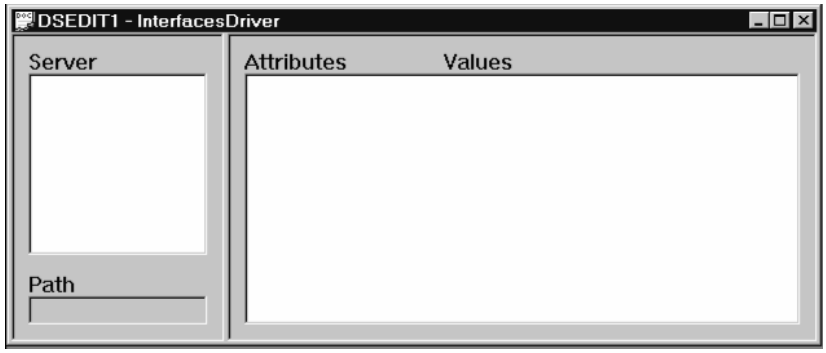
[Select Directory Service] ウィンドウで、ディレクトリ・サービスのセッションを開きます。セッションを開くと、interfaces ファイル (*SQL.ini*)、または *libtcl.cfg* ファイルにリストされたドライバを持つディレクトリ・サービスを編集できます。

❖ **セッションを開くには、次の手順に従います。**

- [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。

  Adaptive Server 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は、*DSEdit* は *libtcl.cfg* ファイルを検索できません。



このウィンドウで、Adaptive Server 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] ー [Add] を選択します。

   [Input Server Name] 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。

サーバ名のエントリは、Adaptive Server Anywhere の起動に使用した名前と一致させます。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。Adaptive Server 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

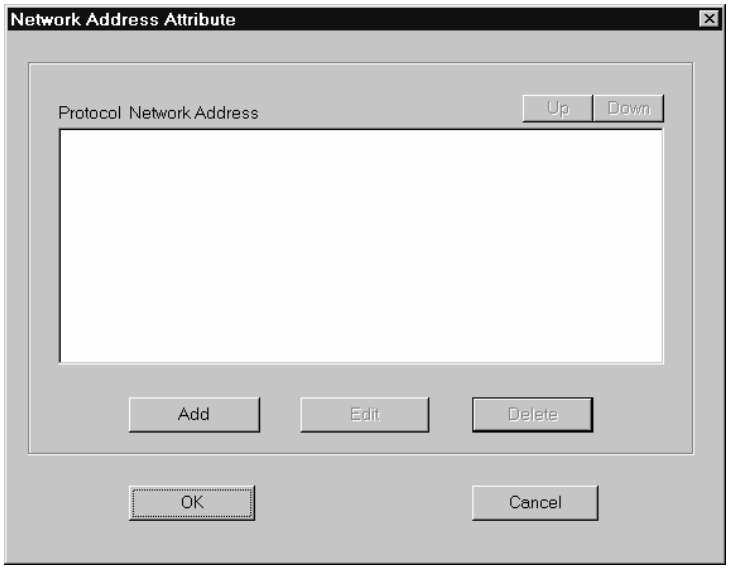
サーバ名を入力した場合、サーバ・アドレスを修正して interfaces ファイル入力を完了する必要があります。

❖ サーバ・アドレスを入力するには、次の手順に従います。

1 [Server] 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。

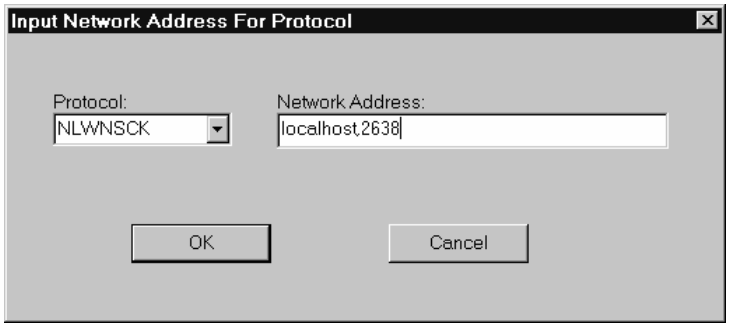
3 ポップアップ・メニューから [Modify Attribute] を選択します。ウィンドウが表示され、アドレスの現在の値を表示します。アドレスを 1 つも入力しなかった場合は、ボックスは空になります。



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] リスト・ボックスから、TCP/IP プロトコルの [NLWNSCK] を選択し、[ネットワーク・アドレス] テキスト・ボックスに値を入力します。



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

サーバが動作しているマシンは、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] でマシン名を検索できます。

クライアントとサーバが同一のマシン上にある場合でも、マシン名を入力します。この場合は、**localhost** を使用して現在のマシンを識別できます。

ポート番号

入力するポート番号は、「データベース・サーバを Open Server として起動する」138 ページで説明するように、Adaptive Server 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。Adaptive Server 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 Adaptive Server 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、指定されたマシン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] ー [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] ー [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。

JDBC URL については、『ASA プログラミング・ガイド』>「サーバの URL の指定」を参照してください。

Open Client と jConnect 接続の特性

Adaptive Server 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

オプション	設定値
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」795 ページを参照してください。

第 5 章

データベース・サーバ

この章の内容

この章では、Adaptive Server Anywhere データベース・サーバのコマンド・ライン・オプションについて説明します。

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

{ **dbeng9** | **dbsrv9** }
[*server-options*] [*database-file* [*database-options*] …]

NetWare を使用している場合の構文

load dbsrv9 [*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出す。「@data サーバ・オプション」164 ページを参照。
-?	使用法を表示する。「-? サーバ・オプション」166 ページを参照。
-b	バルク・オペレーション・モードで実行する。「-b サーバ・オプション」166 ページを参照。
-c *size*	初期キャッシュ・サイズを設定する。「-c サーバ・オプション」167 ページを参照。
-ca 0	動的なキャッシュ・サイズの設定を無効にする (Windows NT/2000/XP、Windows 95/98/Me、UNIX)。「-ca サーバ・オプション」170 ページを参照。
-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する。「-cc サーバ・オプション」170 ページを参照。
-ch *size*	キャッシュ・サイズの上限値を設定する (Windows NT/2000/XP、Windows 95/98/Me)。「-ch サーバ・オプション」171 ページを参照。

サーバ・オプション	説明
-cl *size*	キャッシュ・サイズの下限値を設定する (Windows NT/2000/XP)。「-cl サーバ・オプション」172 ページを参照。
-cr { **+** | **-** }	データベース・ページを保持するキャッシュを準備する。「-cr サーバ・オプション」173 ページを参照。
-cs	データベース・サーバ・ウィンドウにキャッシュの使用状況を表示する。「cs サーバ・オプション」174 ページを参照。
-ct { **+** | **-** }	文字セット変換のオンとオフを切り替える (NetWare や Windows CE は除く)。「-ct サーバ・オプション」174 ページを参照。
-cv { **+** | **-** }	データベース・サーバ・ウィンドウにキャッシュ・ウォーミングに関するメッセージの表示を制御する。「-cv サーバ・オプション」175 ページを参照。
-cw	データベース・サーバ・キャッシュのサイズを設定するために、Windows 2000、Windows XP、Windows Server 2003 での Address Windowing Extensions (AWE) の使用を有効にする。「-cw サーバ・オプション」176 ページを参照。
-d	POSIX I/O [NetWare] を使用する。「-d サーバ・オプション」180 ページを参照。
-ec *encryption-options*	パケットの暗号化を有効にする (ネットワーク・サーバ)。「-ec サーバ・オプション」180 ページを参照。
-ep	暗号化キーを入力するよう要求する。「-ep サーバ・オプション」184 ページを参照。
-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する。「-fc サーバ・オプション」186 ページを参照。

サーバ・オプション	説明
-fips	強力なデータベースおよび通信の暗号化に FIPS 承認のアルゴリズムのみを使用することを要求する。「-fips サーバ・オプション」187 ページを参照。
-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する。さらに、最後のデータベースを閉じた後で停止する (NetWare を除く)。「-ga サーバ・オプション」189 ページを参照。
-gb *level*	データベース・プロセスの優先度クラスを *level* に設定する (Windows NT/2000/XP)。「-gb サーバ・オプション」189 ページを参照。
-gc *num*	最大チェックポイント・タイムアウト時間を *num* 分に設定する。「-gc サーバ・オプション」190 ページを参照。
-gd *level*	データベース起動パーミッションを設定する。「-gd サーバ・オプション」190 ページを参照。
-ge *size*	外部関数を実行するスレッドのスタック・サイズを設定する。「-ge サーバ・オプション」192 ページを参照。
-gf	トリガの起動を無効にする。「-gf サーバ・オプション」192 ページを参照。
-gk *level*	サーバを停止するためのパーミッションを設定する。「-gk サーバ・オプション」192 ページを参照。
-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する。「-gl サーバ・オプション」193 ページを参照。
-gm *num*	接続の最大数を設定する。「-gm サーバ・オプション」194 ページを参照。
-gn *num*	データベース・サーバが一度に処理できる同時要求の最大数を設定する。「-gn サーバ・オプション」194 ページを参照。

サーバ・オプション	説明
-gp *size*	最大ページ・サイズを *size* バイトに設定する。「-gp サーバ・オプション」195 ページを参照。
-gr *minutes*	最大リカバリ時間を *num* 分に設定する。「-gr サーバ・オプション」196 ページを参照。
-gss *size*	スレッドのスタック・サイズを *size* バイトに設定する (Windows には適用されない)。「-gss サーバ・オプション」196 ページを参照。
-gt *num*	データベース・サーバで使用するオペレーティング・プロセッサの数を *num* プロセッサに設定する。「-gt サーバ・オプション」197 ページを参照。
-gu *level*	次のようなユーティリティ・コマンドのパーミッション・レベルを設定する。**utility_db**、**all**、**none**、**DBA**。「-gu サーバ・オプション」197 ページを参照。
-gx	データベース・サーバ・プロセスに割り当てるオペレーティング・システムのスレッド数を設定する [Windows NT/2000/XP、Windows 95/98/Me]。「-gx サーバ・オプション」198 ページを参照。
-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する。「-m サーバ・オプション」199 ページを参照。
-n *name*	データベース・サーバ名を *name* にする。-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」200 ページを参照。
-o *filename*	指定したファイルにメッセージを出力する。「-o サーバ・オプション」202 ページを参照。
-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する。「-oe サーバ・オプション」202 ページを参照。
-os *size*	メッセージ用のログ・ファイルのサイズを制限する。「-os サーバ・オプション」202 ページを参照。

サーバ・オプション	説明
-p *packet-size*	最大ネットワーク・パケット・サイズを設定する (ネットワーク・サーバ)。「-p サーバ・オプション」203 ページを参照。
-pc	同一マシン接続以外のすべての接続のパケットを圧縮する。「-pc サーバ・オプション」203 ページを参照。
-pt *size_in_bytes*	圧縮を適用する最小のネットワーク・パケット・サイズを設定する。「-pt サーバ・オプション」204 ページを参照。
-qi	データベース・サーバ・トレイ・アイコンまたは画面を表示しないようにする。「-qi サーバ・オプション」205 ページを参照。
-qp	データベース・サーバ・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する。「-qp サーバ・オプション」205 ページを参照。
-qs	起動エラー・ダイアログを表示しないようにする。「-qs サーバ・オプション」205 ページを参照。
-qw	データベース・サーバ画面を表示しないようにする。「-qw サーバ・オプション」206 ページを参照。
-r	読み込み専用モードでデータベースを開く。「-r サーバ・オプション」206 ページを参照。
-s	syslog facility ID を設定する (UNIX)。「-s サーバ・オプション」207 ページを参照。
-sb { **0** | **1** }	ブロードキャストに対するサーバの動作を指定する。「-sb サーバ・オプション」209 ページを参照。
-sc	共有メモリ・ポートを無効にして、名前付きパイプを有効にする (Windows NT/2000/XP)。「-sc サーバ・オプション」210 ページを参照。
-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。デフォルト値は 240 分。「-ti サーバ・オプション」210 ページを参照。

サーバ・オプション	説明
-tl *seconds*	デフォルトのクライアント活性タイムアウト (秒数)。デフォルト値は 120 秒。「-tl サーバ・オプション」211 ページを参照。
-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る (Windows NT/2000/XP)。「-tmf サーバ・オプション」212 ページを参照。
-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する (Windows NT/2000/XP)。「-tmt サーバ・オプション」212 ページを参照。
-tq *time*	終了時刻を設定する (ネットワーク・サーバ)。「-tq サーバ・オプション」213 ページを参照。
-u	バッファ・ディスク I/O を使用する。「-u サーバ・オプション」213 ページを参照。
-ua	非同期 I/O の使用をオフにする [Linux]。「-ua サーバ・オプション」214 ページを参照。
-uc	
-ud	デーモンとして実行する (UNIX)。「-ud サーバ・オプション」215 ページを参照。
-ui	
-ut *minutes*	*min* 分ごとにテンポラリ・ファイルにタッチする (UNIX)。「-ut サーバ・オプション」216 ページを参照。
-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] ダイアログを表示する [Linux と Solaris のみ]。「-ux サーバ・オプション」217 ページを参照。
-v	データベース・サーバのバージョンを表示して停止する。「-v サーバ・オプション」218 ページを参照。
-x *list*	通信リンクのリストをカンマで区切って提供する。「-x サーバ・オプション」218 ページを参照。

サーバ・オプション	説明
-xs	サーバ側の Web サービス通信プロトコルを指定する。「-xs サーバ・オプション」220 ページを参照。
-y	Windows 95/98/Me サービスとして実行する (Windows 95/98/Me)。「-y サーバ・オプション」222 ページを参照。
-z	通信リンクに関する診断情報を提供する (ネットワーク・サーバ)。「-z サーバ・オプション」223 ページを参照。
-zl	各接続の最後に作成された SQL 文の取得をオンにする。「-zl サーバ・オプション」223 ページを参照。
-zn	保管する要求ログ・ファイルのコピー数を指定する。「-zn サーバ・オプション」224 ページを参照。
-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する。「-zo サーバ・オプション」225 ページを参照。
-zr { **all** | **SQL** | **none** }	SQL オペレーションのログを有効にする。デフォルトは、NONE です。「-zr サーバ・オプション」226 ページを参照。
-zs *size*	要求ロギング用のログ・ファイルのサイズを制限する。「-zs サーバ・オプション」226 ページを参照。

リカバリ・オプション

リカバリ・オプション	説明
-a *filename*	名前付きトランザクション・ログ・ファイルを適用する。「-a リカバリ・オプション」227 ページを参照。
-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する。「-f リカバリ・オプション」228 ページを参照。

データベース・オプション

データベース・オプション	説明
-ek *key*	暗号化キーを指定する。「-ek データベース・オプション」229 ページを参照。
-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) する。「-m データベース・オプション」230 ページを参照。
-n *name*	データベースに名前を付ける。-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 データベース・オプション」231 ページを参照。
-r	読み込み専用モードで指定のデータベースを開く。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」232 ページを参照。

説明

dbeng9 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。**dbsrv9** 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・サイズについては、「-c サーバ・オプション」167 ページを参照してください。

Windows NT/2000/XP、Windows 95/98/Me、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。

詳細については、『ASA SQL ユーザーズ・ガイド』>「パフォーマンス向上へのキャッシュの使用」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。詳細については、「-ch サーバ・オプション」171 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。詳細については、「-ca サーバ・オプション」170 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・マシンから Sybase Central によってデータベース管理などのタスクを実行することができます。

注意
パフォーマンスと信頼性の向上を確保するために、ネットワーク・データベース・サーバは Windows 95/98/Me ではなく Windows NT/2000/XP または Windows Server 2003 で実行することをおすすめします。

NetWare に関する注意

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* は、データベース・ファイル名を指定します。*database-file* がファイル拡張子なしで指定された場合、Adaptive Server Anywhere はまず拡張子 *.wrt* (ライト・ファイル) の付いた *database-file* を検索し、次に拡張子 *.db* の付いた *database-file* 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。また、UNC (Universal Naming Convention) フォーマットに準拠するパスも指定できます。

```
¥¥server¥volume¥path¥file.ext
```

Windows イベント・ログ・メッセージを出力しない

データベース・サーバを Windows サービスとして実行する場合、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・エントリを出力しないようにできます。レジストリ・エントリは、次のとおりです。

```
Software¥Sybase¥Adaptive Server Anywhere¥9.0
```

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」3 ページ
- 「ネットワーク・プロトコル・オプション」276 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・トレー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・オプションによるエラー・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

機能

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出します。

構文

{ **dbsrv9** | **dbeng9** } @*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、Interactive SQL (dbisql)、言語ユーティリティ (dblang)、Adaptive Server Anywhere Console ユーティリティ (dbconsole)、再構築ユーティリティ (rebuild)、証明書生成ユーティリティ (gencert)、証明書読み込みユーティリティ (readcert)、ActiveSync プロバイダ・インストール・ユーティリティ (dbasinst)、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。

説明

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の詳細については、「設定ファイルの使用」642 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の詳細については、「ファイル非表示ユーティリティ」679 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。

@data パラメータは、-w オプションのあとに開始された場合そのまま使用されます。たとえば、データベース・サーバがパラメータ・ファイル自身を読み込むサービスを作成するのにサービス作成ユーティリティ [dbsvc] を使用する場合に役立ちます。

例

次のファイルは、すべてのパラメータを *dbinit.in* から読み込みます。

```
dbinit @dbinit.in
```

次の例は、`dbinit -q -ja -p 2048 -b new.db` と同じです。

```
echo "-ja -p 2048" > dbinit.in
 dbinit -q @dbinit.in -b new.db
```

次の例では、dbsvc_parms は dbsvc から読み込まれますが、dbsrv_parms は読み込まれません。サービスは、コマンド・ライン `dbsrv9.exe @dbsrv_parms` とともに生成されます。

```
dbsvc @dbsvc_parms -w mysvc dbsrv9.exe @dbsrv_parms
```

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 4MB 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:¥Program Files¥Sybase¥SQL Anywhere 9¥asademo.db"
```

この設定ファイルが *c:¥config.txt* という名前で保存されている場合、コマンドで次のように使用します。

```
dbsrv9 @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 4MB 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:¥Program Files¥Sybase¥SQL
Anywhere 9¥asademo.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。

```
dbsrv9 @envvar
```

-? サーバ・オプション

機能 使用法を表示します。

構文 { **dbsrv9** | **dbeng9** } **-?**

適用対象 すべてのオペレーティング・システムとサーバ

説明 各サーバ・オプションの簡単な説明を表示します。それ以外のタスクは何も実行しません。

-b サーバ・オプション

機能 バルク・オペレーション・モードを使用します。

構文 { **dbsrv9** | **dbeng9** } **-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

説明　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

機能　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　{ **dbsrv9** | **dbeng9** } **-c** { *integer* | *integer***G** | *integer***K** | *integer***M** | *integer***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

説明　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。

単位を表す **G**、**K**、**M** は、大文字と小文字のどちらでも使用できます。**G**、**K**、または **M** が指定されていない場合、10000 未満の整数はキロバイト数と想定され、10000 以上の整数はバイト数と想定されます。たとえば、**-c 4096** は 4096 KB または 4,194,304 バイトを表し、**-c 200 000** は 200,000 バイトの (非常に小さい) キャッシュを意味します。

単位 **P** は物理システム・メモリのパーセンテージを表します。この単位を使用すると、引数はパーセンテージになります。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。物理システム・メモリの 50% を使用するには、次のコマンドを使用します。

```
dbeng9 -c 50%% ...
```

NetWare と UNIX オペレーティング・システムでは、-c で指定されたキャッシュ・サイズが使用可能なメモリ総量より大きい場合、データベース・サーバは、次のように計算した最大キャッシュ・サイズを使用します。

```
95% of (available memory - 5 MB)
```

Windows CE では、-c で指定されたキャッシュ・サイズが使用可能なメモリ総量より大きい場合、データベース・サーバは、次のように計算した最大キャッシュ・サイズを使用します。

```
95% of (available memory - 2 MB)
```

-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** 600K
   - **Windows NT/2000/XP、Windows 95/98/Me、NetWare** 2 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。

   - マシンの物理メモリの 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。

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。

```
dbeng9 -c 3M -n myserver "C:¥Program Files¥Sybase¥SQL
Anywhere 9¥asademo.db"
```

参照

- 「-ch サーバ・オプション」171 ページ

-ca サーバ・オプション

機能 静的キャッシュ・サイズを強制的に適用します。引数 0 は必須です。

構文 { **dbsrv9** | **dbeng9** } **-ca 0** …

適用対象 Windows NT/2000/XP、Windows 95/98/Me、UNIX

説明 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** の設定がない場合、データベース・サーバは必要に応じて自動的にキャッシュを追加します。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や、致命的なエラーを引き起こすほどのメモリが Java VM で必要となる場合、-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。**0** を省略すると、自動キャッシュ増加機能が有効になります。

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。

```
dbsrv9 -c 40P -ca 0 -n myserver "C:¥Program
Files¥Sybase¥SQL Anywhere 9¥asademo.db"
```

参照

- 「-c サーバ・オプション」167 ページ
- 「-ch サーバ・オプション」171 ページ

-cc サーバ・オプション

機能 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 { **dbsrv9** | **dbeng9** } **-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

説明

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」173 ページ
- 「-cv サーバ・オプション」175 ページ
- 『ASA SQL ユーザーズ・ガイド』>「キャッシュ・ウォーミングの使用」

-ch サーバ・オプション

機能

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

{ **dbsrv9** | **dbeng9** } **-ch** { *integer* | *integer***G** | *integer***K** | *integer***M** | *integer***P** }
…

適用対象

Windows NT/2000/XP、Windows 95/98/Me、UNIX

説明

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、256 MB とマシンの物理メモリの 90% のうち、いずれか低い方になります。

単位を表す **G**、**K**、**M** は、大文字と小文字のどちらでも使用できます。**G**、**K**、または **M** が指定されていない場合、10000 未満の整数はキロバイト数と想定され、10000 以上の整数はバイト数と想定されます。たとえば、**-ch 4096** は、4096 KB または 4 194 304 バイトを表し、**-ch 200 000** は 200,000 バイトの (非常に小さい) キャッシュを意味します。

単位 **P** は物理システム・メモリのパーセンテージを表します。この単位を使用すると、引数はパーセンテージになります。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。物理システム・メモリの 50% を使用するには、次のコマンドを使用します。

```
dbeng9 -ch 50%% ...
```

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。

```
dbeng9 -ch 2M -n silver "C:¥Program Files¥Sybase¥SQL
Anywhere 9¥asademo.db"
```

参照

- 「-c サーバ・オプション」167 ページ
- 「-ca サーバ・オプション」170 ページ
- 「-cl サーバ・オプション」172 ページ

-cl サーバ・オプション

機能

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

{ **dbsrv9** | **dbeng9** } **-cl** { *integer* | *integer***G** | *integer***K** | *integer***M** | *integer***P** } …

適用対象

Windows NT/2000/XP、Windows 95/98/Me、UNIX

説明

このオプションは、キャッシュの下限値を設定します。初期のキャッシュ・サイズが、デフォルトの最小キャッシュ・サイズになります。

単位を表す **G**、**K**、**M** は、大文字と小文字のどちらでも使用できます。**G**、**K**、または **M** が指定されていない場合、10000 未満の整数はキロバイト数と想定され、10000 以上の整数はバイト数と想定されます。たとえば、**-cl 4096** は、4096 KB または 4 194 304 バイトを表し、**-cl 200 000** は 200,000 バイトの最小のキャッシュを意味します。

単位 **P** は物理システム・メモリのパーセンテージを表します。この単位を使用すると、引数はパーセンテージになります。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。

```
dbeng9 -cl 50%% ...
```

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、サンプル・データベースをロードします。

```
dbeng9 -cl 5M -n silver "C:¥Program Files¥Sybase¥SQL
Anywhere 9¥asademo.db"
```

参照

- 「-c サーバ・オプション」167 ページ
- 「-ca サーバ・オプション」170 ページ
- 「-ch サーバ・オプション」171 ページ

-cr サーバ・オプション

機能

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

{ **dbsrv9** | **dbeng9** } **-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

説明

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」170 ページ

- 「-cv サーバ・オプション」175 ページ
- 『ASA SQL ユーザーズ・ガイド』>「キャッシュ・ウォーミングの使用」

cs サーバ・オプション

機能 データベース・サーバ・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 { **dbsrv9** | **dbeng9** } **-cs** …

適用対象 動的なキャッシュ・サイズをサポートするすべてのプラットフォーム

説明 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、データベース・サーバ・ウィンドウにキャッシュ情報を表示します。

-ct サーバ・オプション

機能 文字セット変換のオンとオフを切り替えます。

構文 { **dbsrv9** | **dbeng9** } **-ct** { **+** | **-** } …

適用対象 Windows CE を除くすべてのオペレーティング・システム

説明 デフォルトでは、文字セット変換はオンになっています。同じ文字を異なる値で表す文字セット間で、文字列を変換します。これは、クライアント・マシンとデータベースが異なる文字セットを使用している場合に役立ちます。文字セット変換を無効にするには、-ct サーバ・オプションを使用します。この引数を -ct+ として指定すると、文字セット変換がオンになります。

Adaptive Server Anywhere バージョン 7.x 以前では、**+** または **-** 値は使用できません。-ct オプションを指定すると、文字セット変換が有効になります。

例 次の例は、文字セット変換が無効で、サンプル・データベースをロードするサーバを起動します。

```
dbeng9 -ct- "C:¥Program Files¥Sybase¥SQL Anywhere
9¥asademo.db"
```

参照

- 「データベース・サーバで文字セット変換をオフにする」464 ページ
- 「文字セット変換を使用してデータベース・サーバを起動する」464 ページ

-cv サーバ・オプション

機能

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する。

構文

{ **dbsrv9** | **dbeng9** } **-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

説明

-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、データベース・サーバ・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。

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データベース・サーバ・ウィンドウにログします。

```
dbsrv9 -cc+ -cr+ -cv+ mydatabase.db
```

参照

- 「-cc サーバ・オプション」170 ページ
- 「-cr サーバ・オプション」173 ページ
- 『ASA SQL ユーザーズ・ガイド』> 「キャッシュ・ウォーミングの使用」

-cw サーバ・オプション

機能

データベース・サーバ・キャッシュのサイズを設定するために、Windows 2000、Windows XP、Windows Server 2003 での Address Windowing Extensions (AWE) の使用を有効にします。

構文

{ **dbsrv9** | **dbeng9** } **-cw** …

適用対象

Windows 2000、Windows XP、Windows Server 2003 以上

説明

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 2000、Windows XP、Windows Server 2003 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。

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*	8 GB
Windows 2000 Datacenter Server	2.7 GB*	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
Windows Server 2003、Standard Edition	1.8 GB	4 GB
Windows Server 2003、Enterprise Edition	2.7 GB*	32 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、Datacenter Edition	2.7 GB*	64 GB

* 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。

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。

キャッシュ・サイズの指定については、「-c サーバ・オプション」167 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。

- システムで使用可能なメモリが少なくとも 130MB 必要です。
- システムのメモリが 2 GB ～ 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。

  システムのメモリが 16 GB を上回るときは、*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ためです。

- システムのメモリが 4 GB を超える場合、*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。
- 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 2000 での手順を説明します。

  1. Administrator として Windows にログオンします。
  2. [スタート] メニューで [設定] ー [コントロール パネル] を選択します。
  3. [管理ツール] フォルダを開きます。
  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. 左ウィンドウ枠の [ユーザ権限の割り当て] 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。

     [ローカル セキュリティ ポリシーの設定] ダイアログが表示されます。
  8. [ローカル セキュリティ ポリシーの設定] ダイアログで [追加] をクリックします。

     [ユーザまたはグループの選択] ダイアログが表示されます。
  9. リストからユーザ ID を選択して [追加] をクリックします。
  10. [ローカル セキュリティ ポリシーの設定] ダイアログで [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とき、キャッシュ・ページ・サイズは少なくとも 4 KB になり、動的キャッシュ・サイズ決定は無効になります。64 ビットの Windows プラットフォームでは、キャッシュ・ページ・サイズは少なくとも 8 KB です。

動的なキャッシュ・サイズの変更の詳細については、『ASA SQL ユーザーズ・ガイド』> 「パフォーマンス向上へのキャッシュの使用」を参照してください。

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、サンプル・データベースをロードします。

```
dbeng9 -c 12G -cw asademo.db
```

参照

- 「-c サーバ・オプション」167 ページ

-d サーバ・オプション

機能

POSIX I/O を使用します。

構文

{ **dbsrv9** | **dbeng9** } **-d** …

適用対象

NetWare。このオプションは Windows プラットフォームでは推奨されなくなりました。

説明

-d オプションを指定すると、DFS (Direct File System) I/O ではなく POSIX I/O が強制的に使用されます。このオプションは、以前のバージョンの DFS のバグの回避方法として用意されています。POSIX I/O を使用すると非同期 I/O が使用できないことに注意してください。

このオプションは NetWare システム専用です。

-ec サーバ・オプション

機能

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 Adaptive Server 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

{ **dbsrv9** | **dbeng9** } **-ec** *encryption-options* …

encryption-options:

{ **NONE**
  |**SIMPLE**
  |**ECC_TLS (CERTIFICATE=***filename*;
  CERTIFICATE_PASSWORD=*password* **)**
  |**RSA_TLS (CERTIFICATE=***filename*;
  CERTIFICATE_PASSWORD=*password* **)**
  | **RSA_TLS_FIPS (CERTIFICATE=***filename*;
  CERTIFICATE_PASSWORD=*password* **)** } , …

説明

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。

詳細については、『SQL Anywhere Studio セキュリティ・ガイド』>「Adaptive Server Anywhere トランスポート・レイヤ・セキュリティ」を参照してください。

別途ライセンスを取得できるオプションが必要

トランスポート・レイヤ・セキュリティを使用するには、別途ライセンスを取得できる SQL Anywhere Studio セキュリティ・オプションを入手する必要があります。このセキュリティ・オプションは、輸出規制対象品目です。

このコンポーネントを注文するには、『SQL Anywhere Studio の紹介』>「別途ライセンスが入手可能なコンポーネント」を参照してください。

-ec オプションは、いずれかの指定タイプで暗号化された ODBC、OLE DB または Embedded SQL インタフェースからの接続のみを受け入れるようにデータベース・サーバに指示します。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。TDS 接続パラメータを NO に設定すると、これらの暗号化されていない TDS 接続が禁止されます。

詳細については、「TDS プロトコル・オプション」297 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 簡単に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 Adaptive Server 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、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。

ECC_TLS 楕円曲線に基づく Certicom 暗号化テクノロジで暗号化された接続を受け入れます。下位互換性を保つために、ecc_tls を **CERTICOM** と指定することもできます。このタイプの暗号化を使用するには、Solaris、Linux、NetWare、Mac OS X、またはこの暗号化をサポートしている Windows プラットフォーム (Windows CE 以外) で、サーバとクライアントの両方を実行してください。また、接続には TCP/IP ポートを使用します。Solaris、Linux、および Mac OS X 以外の UNIX プラットフォームでは、クライアントまたはサーバの ECC_TLS パラメータは認識されません。このパラメータに指定できる必須の引数は次のとおりです。

- **certificate** サーバ証明書のファイル名。

  これはサーバのプライベート・キーを含んだサーバ証明書です。このサーバ証明書は、自己署名されたものも、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ものもあります。

- **certificate_password** サーバ証明書のパスワードはプライベート・キーです。

ECC_TLS を使用するためには、ECC 暗号化を使用して証明書を生成します。

証明書の作成の詳細については、『SQL Anywhere Studio セキュリティ・ガイド』> 「デジタル証明書の作成」を参照してください。

RSA_TLS RSA に基づく暗号化テクノロジを使用して暗号化された接続を受け入れます。このタイプの暗号化を使用するには、Solaris、Linux、NetWare、Mac OS X、またはこの暗号化をサポートしている Windows プラットフォーム (Windows CE 以外) で、サーバとクライアントの両方を実行してください。また、接続には TCP/IP ポートを使用します。Solaris、Linux、および Mac OS X 以外の UNIX プラット

フォームでは、クライアントまたはサーバの RSA_TLS パラメータは認識されません。このパラメータに指定できる必須の引数は次のとおりです。

- **certificate**　サーバ証明書のファイル名。

  これはサーバのプライベート・キーを含んだサーバ証明書です。このサーバ証明書は、自己署名されたものも、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ものもあります。

- **certificate_password**　サーバ証明書のパスワードはプライベート・キーです。

RSA 暗号化を使用して証明書を生成してください。

証明書の作成の詳細については、『SQL Anywhere Studio セキュリティ・ガイド』>「デジタル証明書の作成」を参照してください。

RSA_TLS_FIPS　FIPS 承認の RSA 暗号化テクノロジで暗号化された接続を受け入れます。RSA_TLS_FIPS は、別の承認済みライブラリを使用しますが、Adaptive Server Anywhere 9.0.2 以降で RSA_TLS を指定するクライアントとの互換性を持っています。このタイプの暗号化を使用するには、サーバとクライアントの両方が、サポートされる 32 ビットの Windows オペレーティング・システムで実行されている必要があります。また、TCP/IP ポートで接続されていることも必要です。このパラメータに指定できる必須の引数は次のとおりです。

- **certificate**　サーバ証明書のファイル名。

  これはサーバのプライベート・キーを含んだサーバ証明書です。このサーバ証明書は、自己署名されたものも、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ものもあります。

- **certificate_password**　サーバ証明書のパスワードはプライベート・キーです。

FIPS 承認の RSA 暗号化を使用する場合は、RSA 暗号化を使用して証明書を生成してください。

証明書の作成の詳細については、『SQL Anywhere Studio セキュリティ・ガイド』>「デジタル証明書の作成」を参照してください。

dbecc9.dll ファイルと *dbrsa9.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 コードが含まれます。*dbrsa9f.dll* には FIPS 承認の RSA アルゴリズムのコードが含まれています。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、メッセージがサーバ・ウィンドウに表示されます。指定されたタイプの暗号化が開始できないと、サーバが開始しません。

クライアントとサーバの暗号化の設定は同じにしてください。異なる場合は接続が失敗します。ただし、サーバで **-ec simple** が指定され、**-ec none** が指定されていない場合は、暗号化を要求しない接続が許可され、単純暗号化が自動的に使用されます。

例

次の例は、dbsrv9 コマンド・ラインで楕円曲線サーバ証明書 *sample.crt* を指定しています。

```
dbsrv9 -ec ecc_tls(certificate=sample.crt;
certificate_password=tJ1#m6+W) -x tcpip asademo.db
```

次の例は、dbsrv9 コマンド・ラインで RSA サーバ証明書 *rsaserver.crt* を指定しています。

```
dbsrv9 -ec rsa_tls(certificate=rsaserver.crt;
certificate_password=test) -x tcpip asademo.db
```

次の例は、dbsrv9 コマンド・ラインで RSA サーバ証明書 *rsaserver.crt* を指定しています。rsa_tls_fips パラメータは、FIPS 承認の RSA アルゴリズムに使用されます。

```
dbsrv9 -ec rsa_tls_fips(certificate=rsaserver.crt;
certificate_password=test) -x tcpip asademo.db
```

参照

- 『SQL Anywhere Studio セキュリティ・ガイド』> 「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」
- 「Encryption 接続パラメータ [ENC]」256 ページ

-ep サーバ・オプション

機能

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

{ **dbsrv9** | **dbeng9** } **-ep** …

説明

-ep オプションは、ユーザが暗号化キーを入力するためのダイアログ・ボックスを表示するようにデータベース・サーバに指示します。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。

サポートされているツールと一緒に使用すると、暗号化キーが必要でない場合でも、常にキーを要求します。ダイアログ・ボックスでプロンプト画面が表示されても、キーが必要でないとわかっている場合は、[キャンセル] をクリックして続行できます。

サーバで使用すると、ユーザに暗号化キーを要求するのは、次の条件がすべてtrue の場合のみです。

- -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。

詳細については、『SQL Anywhere Studio セキュリティ・ガイド』>「Adaptive Server Anywhere トランスポート・レイヤ・セキュリティ」を参照してください。

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。

```
dbsrv9 -ep -x tcpip myencrypted.db
```

参照

- 「-ek データベース・オプション」229 ページ
- 「DatabaseKey 接続パラメータ [DBKEY]」250 ページ

-fc サーバ・オプション

関数

ファイル・システム・フルのコールバック関数を含む 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

{ **dbsrv9** | **dbeng9** } **-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

説明

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。Adaptive Server Anywhere データベース・サーバが DLL とエントリ・ポイントの両方を見つけることができない場合、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¥Asa¥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。

Adaptive Server Anywhere は、その他の DLL やファイルを検索するのと同じ場所でコールバック関数を検索します。

Adaptive Server Anywhere がファイルを検索する場所については、「Adaptive Server Anywhere のファイル検索方法」360 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限り、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ときにディスク・フルの状態になると、致命的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することができます。

詳細については、『ASA SQL リファレンス・マニュアル』>「CREATE EVENT 文」を参照してください。

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。

```
dbeng9 -fc diskfull.dll
```

参照

- 『ASA プログラミング・ガイド』>「コールバック関数の使い方」
- 「システム・イベントの選択」401 ページ
- 「TEMP_SPACE_LIMIT_CHECK オプション [データベース]」901 ページ

-fips サーバ・オプション

関数

強力なデータベースおよび通信の暗号化に 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

{ **dbsrv9** | **dbeng9** } **-fips** …

説明

このオプションを指定すると、すべてのサーバ暗号で 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、および 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。

別途ライセンスを取得できるオプションが必要

トランスポート・レイヤ・セキュリティ、および AES_FIPS を使用した強力なデータベース暗号化を使用するには、別途ライセンスを取得できる SQL Anywhere Studio セキュリティ・オプションを入手する必要があります。このセキュリティ・オプションは、輸出規制対象品目です。

このコンポーネントを注文するには、『SQL Anywhere Studio の紹介』>「別途ライセンスが入手可能なコンポーネント」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マシンには、SQL Anywhere Studio セキュリティ・オプションをインストールしてください。

Adaptive Server 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_TLS が指定されていてもサーバは RSA_TLS_FIPS を使用します。ECC_TLS 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。

参照

- 『SQL Anywhere Studio セキュリティ・ガイド』> 「高度な暗号化」
- 『SQL Anywhere Studio セキュリティ・ガイド』> 「Adaptive Server Anywhere トランスポート・レイヤ・セキュリティ」
- 『SQL Anywhere Studio セキュリティ・ガイド』> 「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」

-ga サーバ・オプション

機能　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　{ **dbsrv9** | **dbeng9** } **-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

説明　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パーソナル・サーバも停止します。

-gb サーバ・オプション

機能　データベース・プロセスの優先クラスを設定します。

構文　{ **dbsrv9** | **dbeng9** } **-gb** { **idle** | **normal** | **high** | **maximum** } …

適用対象　Windows NT/2000/XP

説明　データベース・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

機能

チェックポイント間の期待最大間隔を設定します。

構文

{ **dbsrv9** | **dbeng9** } **-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

説明

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。

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、このオプションによって上書きされないかぎり、最初に起動するデータベースによって指定されたチェックポイント時間が使用されます。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。

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」518 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]」829 ページ
- 「チェックポイントとチェックポイント・ログ」514 ページ

-gd サーバ・オプション

機能

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

{ **dbsrv9** | **dbeng9** } **-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

説明

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・サーバは **DBA** です。構文には、大文字と小文字のいずれも使用できます。

このオプションが DBA に設定された場合、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 ユーザ名とパスワードを指定するだけでは不十分です。

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. データベース・サーバ実行プログラムを保持するディレクトリにある *util_db.ini* ファイルに、パスワードを入力します。

   ```
   [UTILITY_DB]
    pwd=mypwd
   ```

2. ネットワーク・データベース・サーバを起動します。

   ```
   dbsrv9 -x tcpip -n myserver -gd DBA
   ```

3. Interactive SQL からユーティリティ・データベースに接続します。

   次のコマンドは、**myserver** がデフォルトのデータベース・サーバであることを想定しています。

   ```
   dbisql -c "uid=DBA;pwd=mypwd;dbn=utility_db "
   ```

4. データベースを起動します。

   ```
   start database asademo
    on myserver;
   ```

5. 起動したデータベースに接続します。

```
connect
 to myserver
 database asademo
 user DBA identified by SQL
```

-ge サーバ・オプション

機能 外部関数のスタック・サイズを設定します。

構文 { **dbsrv9** | **dbeng9** } **-ge** *integer* …

適用対象 Windows 95/98/Me、Windows NT/2000/XP、NetWare

説明 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 32K。

-gf サーバ・オプション

機能 サーバ上でトリガを起動不可にします。

構文 { **dbsrv9** | **dbeng9** } **-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

説明 -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]」 847 ページ

-gk サーバ・オプション

機能 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 { **dbsrv9** | **dbeng9** } **-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

説明 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。

構文には、大文字と小文字のいずれも使用できます。

-gl サーバ・オプション

機能

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

{ **dbsrv9 | dbeng9** } **-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

説明

UNLOAD TABLE や UNLOAD 文は、データベース・サーバ・マシン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・マシン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。

構文には、大文字と小文字のいずれも使用できます。

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。これらの設定には、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・サーバが現在のマシンで実行され、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が反映されています。

-gm サーバ・オプション

機能 サーバに対する同時接続の数を制限します。

構文 { **dbsrv9** | **dbeng9** } **-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

説明 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

機能 データベース・サーバが同時に処理できる (ユーザとシステムの) アクティブな要求の最大数を設定します。

構文 { **dbsrv9** | **dbeng9** } **-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

説明 データベース・サーバが同時に処理できるアクティブなユーザとシステムの要求の最大数を設定します。データベース・サーバが追加の要求を受け取るときにアクティブな要求の最大数をすでに使用している場合、新しい要求は他の要求が完了するまで待機します。

各接続では、1 つの要求に対して 1 つのスレッドを使用します。要求を処理すると、スレッドは他の接続が使用できるようプールに戻ります。接続は同時に複数の要求を処理できないので、接続が同時に複数のスレッドを使用することはありません。この規則の例外として、Java アプリケーションがスレッドを使用する場合があります。Java アプリケーションでは、各スレッドがデータベース・サーバで使われる実行スレッドとなります。

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・サーバのデフォルトのスレッド数は 20 個ですが、Windows CE のデフォルトは 3 個です。

Windows NT/2000/XP では、パフォーマンス・モニタの [**要求： アクティブ**] と [**要求： 未スケジュール**] の値を調べることができます。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 を増やすことができます。パフォーマンス・モニタは、UNIX または Linux プラットフォームでは使用できません。

参照

- 「コマンド・ラインからスレッドを制御する」17 ページ

-gp サーバ・オプション

機能

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

{ **dbsrv9** | **dbeng9** } **-gp** { **1024** | **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

説明

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。

すべての UNIX プラットフォームでは、最小のページ・サイズは 2048 バイトです。これより小さいページ・サイズのデータベースも使用できますが、キャッシュ・メモリの使用効率が悪くなります。このスイッチを指定しないで、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 2048 になります。

他のプラットフォームでは、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 1024 になります。

-gr サーバ・オプション

機能

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

{ **dbsrv9** | **dbeng9** } **-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

説明

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。

詳細については、「RECOVERY_TIME オプション [データベース]」890 ページ を参照してください。

-gss サーバ・オプション

機能

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

{ **dbsrv9** | **dbeng9** } **-gss** { *integer* | *integer***K** | *integer***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。

説明

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。

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のスタック・サイズは 64 KB、最小スタック・サイズは 500 KB、最大スタック・サイズは 4 MB です。

参照

- 「コマンド・ラインからスレッドを制御する」17 ページ

-gt サーバ・オプション

機能

データベース・サーバで同時に実行できる要求の最大数を設定します。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

{ **dbsrv9** | **dbeng9** } **-gt** *integer* …

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

説明

per-seat ライセンスの場合、ネットワーク・データベース・サーバはマシン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。

参照

- 「コマンド・ラインからスレッドを制御する」17 ページ

-gu サーバ・オプション

機能

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

{ **dbsrv9** | **dbeng9** } **-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

説明

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、次のいずれかに設定できます。**utility_db**、**all**、**none**、**DBA**。デフォルトは **DBA** です。

utility_db レベルは、このコマンドの使用をユーティリティ・データベースに接続できるユーザだけに許可します。ユーティリティ・コマンドの実行を、**all** レベルはすべてのユーザに許可、**none** レベルはすべてのユーザに不許可、**DBA** は DBA 権限を持つユーザに対して許可します。

-gx サーバ・オプション

機能

システム・コールのブロックを同時に実行できる要求の最大数を設定します。

構文

{ **dbsrv9** | **dbeng9** } **-gx** *integer* …

適用対象

Windows 95/98/Me、Windows NT/2000/XP

説明

このオプションは、Windows NT でのみ使用されます。他のプラットフォームでは、-gx 値は、-gt 値に相当します。

有効な値は、-gt オプション値と -gn オプション値で指定した範囲です。デフォルトでは、このオプションはマシン上の CPU 数よりも 1 つ大きい値に設定されます。追加のプロセスにより、1 つのデータベース・サーバ上にある複数のデータベース間のリモート・データ・アクセスが許可されます。

-gx を増やすことにより、主にキャッシュ用に使用できる (物理メモリではなく) アドレス領域が減ります。このオプションは、リモート・データ・アクセス、Java、または外部ストアド・プロシージャ (Java ／外部ストアド・プロシージャの場合はこれらの機能がサーバを実装するために使用されているとき) を使用している場合でのみ使用されます。リモート・データ・アクセス／ Java ／外部ストアド・プロシージャを使用していて、原因不明の理由でサーバがハングした場合、-gx を増やすと問題を解決できる場合があります。-gx オプションを -gn オプション以上の値に設定しても効果がありません。

オプションの設定をデフォルトよりも増やして、標準のデータベース・タスクとは別の外部タスクのために容量を確保することもできます。たとえば、リモート・データ・アクセスを使用する各同時接続に対して -gx 値を増やせます。または、外部ポートで受信するために

データベースで Java を使用する接続に対しても増やせます。これ以外の場合は、このオプションはデフォルト値のままにしておいてください。

UNIX では、各タスクは専用のスレッドで実行されるため、タスク数 (-gn) はスレッド数も決定します。

参照

- 「コマンド・ラインからスレッドを制御する」17 ページ

-m サーバ・オプション

機能

チェックポイントの実行後にトランザクション・ログを削除します。

構文

{ **dbsrv9** | **dbeng9** } **-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

説明

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。-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 文および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。

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

機能

データベース・サーバの名前を設定します。

構文

{ **dbsrv9** | **dbeng9** } **-n** *database-file-name…*

適用対象

すべてのオペレーティング・システムとサーバ

説明

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データベース・ファイル名を受け取ります。たとえば、サーバがファイル *c:¥Program Files¥Sybase¥SQL Anywhere 9¥asademo.db* 起動するときに -n オプションを指定しないと、サーバの名前は **asademo** 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、サーバ名はマシンの文字セットに基づいて解釈されます。サーバ名には、マルチバイト文字を使用しないでください。

この名前は、有効な識別子である必要があります。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
UNIX 共有メモリ	31 バイト

プロトコル	トランケーションの長さ
UNIX 以外の共有メモリ	40 バイト
TCP/IP	40 バイト
SPX	32 バイト
名前付きパイプ	8 バイト

次のようなものはデータベース・サーバ名に使用できません。

- 空白スペース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

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、サーバ SERV とデータベース DATA が指定されます。

```
dbsrv9 -n SERV asademo.db -n DATA
```

詳細については、「-n データベース・オプション」231 ページを参照してください。

参照

- 『ASA SQL リファレンス・マニュアル』> 「識別子」

- 「EngineName 接続パラメータ [ENG]」261 ページ

-o サーバ・オプション

機能 すべてのサーバ・ウィンドウ出力をファイルに出力します。

構文 { **dbsrv9** | **dbeng9** } **-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

説明 すべてのサーバ・メッセージ・ウィンドウの出力をファイルに出力します。次のコマンドを実行すると、ファイル名を取得できます。

```
SELECT property ( 'ConsoleLogFile' )
```

-oe サーバ・オプション

機能 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名を指定します。

構文 { **dbsrv9** | **dbeng9** } **-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

説明 このファイルの各行には、日付と時間がプレフィックスとして付けられます。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

-oe が指定されているか否かに関わらず、致命的なエラーおよびアサーションが 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。

-os サーバ・オプション

機能 サーバ・ウィンドウ出力のファイル・サイズを制限します。

構文　{ **dbsrv9** | **dbeng9** } **-os** { *integer* | *integer***G** | *integer***K** | *integer***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

説明　-o オプションで使用されるログ・ファイルのサイズを制限します。単位を表す **G**、**K**、**M** は、大文字と小文字のどちらでも使用できます。**G**、**K**、または **M** が指定されていない場合、10000 未満の整数はキロバイト数と想定され、10000 以上の整数はバイト数と想定されます。

参照
- 「-o サーバ・オプション」202 ページ

-p サーバ・オプション

機能　通信パケットの最大サイズを設定します。

構文　{ **dbsrv9** | **dbeng9** } **-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

説明　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]」241 ページ

-pc サーバ・オプション

機能　同一マシン接続以外のすべての接続を圧縮します。

構文　{ **dbsrv9** } **-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

説明　Adaptive Server 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、同一マシン接続と TDS 接続を除くすべての接続が圧縮されます。TDS 接続 (jConnect を含む) では、Adaptive Server 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、同一マシン接続は -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できません。

参照

- 「パフォーマンス改善のための通信圧縮設定の調整」125 ページ
- 「Compress 接続パラメータ [COMP]」245 ページ
- 『ASA SQL ユーザーズ・ガイド』> 「Adaptive Server Anywhere の圧縮機能の使用」

-pt サーバ・オプション

機能

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

{ **dbsrv9** } **-pt** *size_in_bytes* ...

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

説明

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することができるようになり、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」125 ページ
- 「CompressionThreshold 接続パラメータ [COMPTH]」247 ページ
- 『ASA SQL ユーザーズ・ガイド』> 「Adaptive Server Anywhere の圧縮機能の使用」

-qi サーバ・オプション

機能　データベース・サーバ・トレイ・アイコンまたはウインドウを表示するか否かを制御します。

構文　{ **dbsrv9** | **dbeng9** } **-qi** …

適用対象　Windows プラットフォーム (Windows CE 以外)

説明　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。

-qp サーバ・オプション

機能　データベース・サーバ・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。

構文　{ **dbsrv9** | **dbeng9** } **-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

説明　データベース・サーバ・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

機能　起動エラー・ダイアログを表示しません。

構文　{ **dbsrv9** | **dbeng9** } **-qs** …

適用対象　Windows のみ

説明

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。これらのエラーは -o または -oe ログには記録されませんが、Windows アプリケーション・イベント・ログ (Window CE を除く) に記録されます。

-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

機能

データベース・サーバ画面を表示しません。

構文

{ **dbsrv9 | dbeng9** } **-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

説明

このオプションは、データベース・サーバ・ウィンドウ (Windows プラットフォーム) とコンソール上の表示メッセージ (非 Windows プラットフォーム) を表示しなくなるようにするものです。

-r サーバ・オプション

機能

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　{ **dbsrv9** | **dbeng9** } **-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

説明　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。

変更できないデータベース・ファイルの例として、CD-ROM デバイスで配布されているデータベースや圧縮されているデータベースなどが挙げられます。ライト・ファイルを作成してデータベース・ファイル以外のデータベースに変更を加えるか、読み込み専用モードで実行することはできます。

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。

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。

```
dbeng9 -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。

```
dbeng9 database1.db -r database2.db
```

-s サーバ・オプション

機能　syslog メッセージのユーザ ID を設定します。

構文 { **dbsrv9** | **dbeng9** } **-s** { **none** | **user** | **daemon** | **local*n*** } …

適用対象 UNIX

説明 **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 (たとえば、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、または -ud データベース・サーバ・オプションで起動した場合の) サーバのデフォルトは **daemon** です。

none の値を指定すると、syslog メッセージはログに記録されません。**local*n*** 引数を使用すると、ファイルへのメッセージのリダイレクトに機能識別子を使用できます。***n*** に 0 ～ 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、および Mac OS X でも使用できます。DEC や HP などの他のプラットフォームでは、*syslog.conf* ファイルが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* ファイルを編集して、次の行を追加します。ここで、**local*n*** は手順 1 で選択した機能識別子です。

   local*n*`.err;`**local*n***`.info;`**local*n***`.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。ここで *<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 Adaptive Server Anywhere データベース・サーバを起動します。ここで、**local*n*** は手順 1 で選択した機能識別子です。

```
dbeng9 -s local n ...
```

これで、Adaptive Server Anywhere データベース・サーバが syslog にレポートするメッセージが、*/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

機能　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　{ **dbsrv9 | dbeng9** } **-sb** { **0 | 1** } ...

適用対象　SPX、TCP/IP

説明　**-sb 0** を使用すると、サーバで TCP/UDP 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、クライアントは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ことになります。また、dblocate を使用したときに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** や **ENG=<*name*>** を指定するとサーバに接続できます。

-sc サーバ・オプション

機能 共有メモリ通信プロトコルを無効にして、名前付きパイプを使用します。

構文 { **dbsrv9** | **dbeng9** } **-sc** …

適用対象 Windows NT/2000/XP

説明 このオプションは、同一マシン通信で使用される共有メモリ通信プロトコルを無効にし、NamedPipes プロトコルを有効にします。

このオプションは、セキュリティ証明を取得するためのイニシアティブの一部として実装されています。このオプションを日常的に使うのは、C2 準拠の環境で実行する場合だけです。

-ti サーバ・オプション

機能 非アクティブ接続を切断します。

構文 { **dbsrv9** | **dbeng9** } **-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

説明 *minutes* で指定された分数の間、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・マシンは、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 *dbsrv9* 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 *dbeng9* 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

- 『ASA SQL リファレンス・マニュアル』> 「sa_server_option システム・プロシージャ」

-tl サーバ・オプション

機能

活性パケットを送信する期間を設定します。

構文

{ **dbsrv9** | **dbeng9** } **-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

参照

- 『ASA SQL リファレンス・マニュアル』> 「sa_server_option システム・プロシージャ」

説明

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、マシンのハードウェア、マシンの 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。

```
dbsrv9 -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。

```
dbsrv9 -tl 3600
```

-tmf サーバ・オプション

機能 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 { **dbsrv9** | **dbeng9** } **-tmf** …

適用対象 Windows NT/2000/XP のみ

説明 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、分散トランザクションのリカバリに使用します。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・ログにある分散トランザクションを持つデータベースを起動する場合にも使用されます。

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。日常的に使用するオプションではありません。

-tmt サーバ・オプション

機能 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 { **dbsrv9** | **dbeng9** } **-tmt** *milliseconds* …

適用対象 Windows NT/2000/XP のみ

説明　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)。

-tq サーバ・オプション

機能　指定の時刻にサーバを停止します。

構文　{ **dbsrv9** | **dbeng9** } **-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

説明　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」483 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

- 『ASA SQL リファレンス・マニュアル』> 「sa_server_option システム・プロシージャ」

-u サーバ・オプション

機能　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　{ **dbsrv9** | **dbeng9** } **-u** …

適用対象　Windows NT/2000/XP、Windows 95/98/Me、UNIX

説明　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。

サーバを専用マシンで実行している場合は、-u オプションを使わないでください。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、サーバを実行するマシン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

機能 非同期 I/O の使用をオフにします。

構文 { **dbsrv9** | **dbeng9** } **-ua** ...

適用対象 Linux

説明 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

関数 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 { **dbsrv9** | **dbeng9** } **-uc**

適用対象 UNIX

説明 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。

デーモンとしてのデータベース・サーバの起動については、「-ud サーバ・オプション」215 ページを参照してください。

参照

- 「-ui サーバ・オプション」215 ページ
- 「-ux サーバ・オプション」217 ページ

例

-ud サーバ・オプション

機能　デーモンとして実行します。

構文　{ **dbsrv9** | **dbeng9** } **-ud** …

適用対象　UNIX

説明　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することができます。

-ui サーバ・オプション

関数　[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 と Solaris 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。

構文　{ **dbsrv9** | **dbeng9** } **-ui**

適用先　X Windows Server がサポートされている Linux と Solaris

説明　-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。

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 のほかにもサーバ・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。

-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・サーバを起動しないようにするには、-ux オプションを指定する必要があります。-uc、-ui、-ux のうち 1 つだけを指定してください。

-ui が指定されているときは、コマンド・ラインの末尾にアンパサンド (&) を追加することで、データベース・サーバをバックグラウンド・タスクとして起動できます。次に例を示します。

```
dbeng9 -ui -n test sample.db &
```

バックグラウンド・タスクはそれを起動したセッションが終わると終了します。-ui が指定されていないときは、バックグラウンド・タスクとして起動されたデータベース・サーバは中断されます。

デーモンとしてのデータベース・サーバの起動については、「-ud サーバ・オプション」215 ページを参照してください。

参照

- 「-uc サーバ・オプション」214 ページ
- 「-ux サーバ・オプション」217 ページ

-ut サーバ・オプション

機能

テンポラリ・ファイルをタッチします。

構文

{ **dbsrv9** | **dbeng9** } **-ut** *minutes* …

適用対象

UNIX

説明

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

関数

[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 と Solaris の場合) を表示します。

構文

{ **dbsrv9** | **dbeng9** } **-ux**

適用対象

X Windows Server がサポートされている Linux と Solaris

説明

-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 オプションしか指定されていない場合は、[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 のほかにもサーバ・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。

-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。

デーモンとしてのデータベース・サーバの起動については、「-ud サーバ・オプション」215 ページを参照してください。

参照

- 「-uc サーバ・オプション」214 ページ
- 「-ux サーバ・オプション」217 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。

```
dbeng9 -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。

```
dbeng9 -ux sample.db
```

-v サーバ・オプション

機能 ソフトウェアのバージョンを表示します。

構文 { **dbsrv9** | **dbeng9** } **-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

説明 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

機能 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 **dbsrv9 -x** { **all** | **none** | *srv-protocols* } …

srv-protocols:
{ [**namedpipes** | **spx** | **tcpip**] *parmlist* },…
parmlist:
(*parm=value*;…**)**

構文 2 **dbeng9 -x** { **all** | **none** | *eng-protocols* } …

eng-protocols:
{ **namedpipes** | **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

説明 -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。

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。
- **NamedPipes (NP)**　NamedPipes 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。NamedPipes は、同一マシン通信のもう 1 つの手段として Windows NT/2000/XP でサポートされています。
- **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、Windows NT/2000/XP、Windows 95/98/Me 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、同一マシン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。

詳細については、「ServerPort プロトコル・オプション [PORT]」294 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説明については、「ネットワーク・プロトコル・オプション」276 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;...)"
```

例

次の場合は、共有メモリ、TCP/IP、SPX の通信だけが許可されます。

```
-x tcpip,spx
```

参照

- 「CommLinks 接続パラメータ [LINKS]」243 ページ

-xs サーバ・オプション

機能

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

{ **dbeng9** | **dbsrv9** } **-xs** { **none** | *web-protocols* } …

web-protocols :
　{ **http** (*parmlist*) | **https** (*parmlist*) |**https_fips** (*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

説明

-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、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。

トランスポート・レイヤ・セキュリティには FIPS 承認の HTTPS_FIPS プロトコルを使用できます。

詳細については、『SQL Anywhere Studio セキュリティ・ガイド』>「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」を参照してください。

別途ライセンスを取得できるオプションが必要
トランスポート・レイヤ・セキュリティを使用するには、別途ライセンスを取得できる SQL Anywhere Studio セキュリティ・オプションを入手する必要があります。このセキュリティ・オプションは、輸出規制対象品目です。

このコンポーネントを注文するには、『SQL Anywhere Studio の紹介』>「別途ライセンスが入手可能なコンポーネント」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ブロードキャストを受信します。次のいずれかを指定できます。

- **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_FIPS** 暗号化に HTTPS プロトコルと FIPS 承認のアルゴリズムを使用するクライアントによる Web 要求を受信します。HTTPS_FIPS は、承認された別個のライブラリを使用しますが、HTTPS との互換性があります。受信するデフォルトのポートは 443 です。

注意
https と https_fips を同時に開始する場合、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 つのポートを変更してください。

- **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説明については、「ネットワーク・プロトコル・オプション」276 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(PARM1=value1;PARM2=value2;...)"
```

例 HTTP Web 要求をポート 80 で受信します。

```
dbeng9 web.db -xs http(port=80)
```

-y サーバ・オプション

機能 サーバを Windows サービスとして実行します。

構文 { **dbsrv9** | **dbeng9** } **-y** ...

適用対象 Windows 95/98/Me

説明 サーバが Windows サービスとして登録されている場合、ユーザがログオンまたはログオフしても稼働し続け、シャットダウン・コマンドも無視されます。

-z サーバ・オプション

機能　トラブルシューティングの目的で、起動時の通信操作を表示します。

構文　{ **dbsrv9** | **dbeng9** } **-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

説明　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、データベース・サーバ・ウィンドウに表示されます。

-zl サーバ・オプション

機能　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　{ **dbsrv9** | **dbeng9** } **-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

説明　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。ある接続について最後に準備された SQL 文を、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。

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 サーバ設定がオンになっている場合、ユーザはだれでも 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することにより、他のユーザが最後に準備した SQL 文を見つけることができます。このオプションの使用には注意が必要で、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」923 ページ

- 『ASA SQL リファレンス・マニュアル』> 「sa_conn_activity システム・プロシージャ」
- 『ASA SQL リファレンス・マニュアル』> 「sa_server_option システム・プロシージャ」

-zn サーバ・オプション

関数

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

{ **dbsrv9** | **dbeng9** } **-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

説明

要求ロギングが長期間有効になっている場合、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することができます。

-z オプションの詳細については、「-zs サーバ・オプション」226 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
- ファイル *req.out.2* ～ *req.out.5* が *req.out.1* ～ *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することができます。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)
```

例

次の例 (すべて 1 行で入力) 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。最大サイズが 10 KB のこのファイルには、要求ログの 3 つのコピーが保存されます。

```
dbeng9 "C:¥Program Files¥Sybase¥SQL Anywhere
9¥asademo.db" -zr all -zn 3
  -zs 10 -zo mydatabase.log
```

参照

- 「-zo サーバ・オプション」225 ページ
- 「-zr サーバ・オプション」226 ページ
- 「-zs サーバ・オプション」226 ページ
- 『ASA SQL リファレンス・マニュアル』> 「sa_server_option システム・プロシージャ」

-zo サーバ・オプション

機能

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

{ **dbsrv9** | **dbeng9** } **-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

説明

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。このファイルから -zo オプションで指定したものとは別のファイルに出力を送信できます。

また、このオプションにより、要求ロギングがコンソールに表示されなくなります。

参照

- 「-zn サーバ・オプション」224 ページ
- 「-zr サーバ・オプション」226 ページ

- 「-zs サーバ・オプション」226 ページ

-zr サーバ・オプション

機能

操作の要求ロギングを有効にします。

構文

{ **dbsrv9** | **dbeng9** } **-zr** { **all** | **SQL** | **none** | **SQL+hostvars** } …

適用対象

すべてのオペレーティング・システムとサーバ

説明

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。この情報はデータベース・サーバ・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。

参照

- 『ASA SQL リファレンス・マニュアル』> 「sa_server_option システム・プロシージャ」
- 「-zn サーバ・オプション」224 ページ
- 「-zo サーバ・オプション」225 ページ

-zs サーバ・オプション

機能

要求ロギング用のファイルのサイズを制限します。

構文

{ **dbsrv9** | **dbeng9** } **-zs** { *integer* | *integer***G** | *integer***K** | *integer***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

説明

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することができます。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。

単位を表す **G**、**K**、**M** は、大文字と小文字のどちらでも使用できます。**G**、**K**、または **M** が指定されていない場合、10000 未満の整数はキロバイト数と想定され、10000 以上の整数はバイト数と想定されます。

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)。その後、要求ログ・ファイルが再起動されます。

参照

- 「-zn サーバ・オプション」224 ページ
- 「-zo サーバ・オプション」225 ページ
- 「-zr サーバ・オプション」226 ページ
- 『ASA SQL リファレンス・マニュアル』> 「sa_server_option システム・プロシージャ」

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。

```
dbeng9 -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 K に到達すると、既存の *mydatabase.old* ファイルが削除され、*mydatabase.log* は *mydatabase.old* と名前変更されて、新しい *mydatabase.log* ファイルが起動されます。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 K) に到達するたびに繰り返されます。

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。

-a リカバリ・オプション

機能　指定したトランザクション・ログを適用します。

構文　{ **dbsrv9** | **dbeng9** } **-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

説明

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)。

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。

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」483 ページを参照してください。

例

次の例 (すべて 1 行に入力) では、ログ・ファイル *asademo.log* がサンプル・データベースに適用されます。

```
dbeng9 "C:¥Program Files¥Sybase¥SQL Anywhere
9¥asademo.db" -a "C:¥backup¥asademo.log"
```

-f リカバリ・オプション

機能

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

{ **dbsrv9** | **dbeng9** } **-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

説明

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。

リカバリ時には、運用で使用するサーバを使用してください。たとえば、運用時にネットワーク・データベース・サーバ (*dbsrv9.exe*) を使用する場合は、リカバリでもそのネットワーク・サーバを使用してく

ださい。これは、パーソナル・サーバの接続制限 (10 件) によって問題が発生することがあるためです。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。

詳細については、「バックアップとデータ・リカバリ」483 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・サーバにサンプル・データベースのリカバリを開始し、実行します。

```
dbeng9 "c:¥Program Files¥Sybase¥SQL Anywhere
9¥asademo.db" -f
```

データベース・オプション

次のオプションは、データベース名の後に入力し、そのデータベースにだけ適用します。

-ek データベース・オプション

機能

強力に暗号化されたデータベースのキーを指定します。

構文

{ **dbsrv9** | **dbeng9** } [*server-options*] *database-file* **-ek** *key* ...

説明

-ek データベース・オプションは、コマンド・ラインでデータベース・ファイル名の後に指定します。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、英数字と特殊文字からなる大文字と小文字が混在する文字列です。

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。

詳細については、「-ep サーバ・オプション」184 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。

詳細については、『SQL Anywhere Studio セキュリティ・ガイド』>「Adaptive Server Anywhere トランスポート・レイヤ・セキュリティ」を参照してください。

例

次の例では、サンプル・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。

```
dbsrv9 -x tcpip asademo.db -ek "Akmm9u70y"
```

参照

- 「-ep サーバ・オプション」184 ページ
- 「DatabaseKey 接続パラメータ [DBKEY]」250 ページ

-m データベース・オプション

機能

チェックポイントの実行後にトランザクション・ログをトランケートします。

構文

{ **dbsrv9** | **dbeng9** } [*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

説明

停止時、またはサーバによってスケジュールされたチェックポイントの結果として、チェックポイント実行後に、トランザクション・ログをトランケート (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 (コマンド・ラインで定義可能) で制御されています。

-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、サンプル・データベースがロードされます。チェックポイントが終了すると、トランザクション・ログがトランケートされます。

```
dbsrv9 -n silver "c:¥Program Files¥Sybase¥SQL Anywhere
9¥asademo.db" -m
```

-n データベース・オプション

機能

データベースの名前を設定します。

構文

{ **dbsrv9** | **dbeng9** } [*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

説明

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 *c:¥asa¥asademo.db* を起動する場合、データベース名は **asademo** になります。

次のようなものはデータベース・サーバ名に使用できません。

- 空白スペース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、サンプル・データベースがロードされます。サンプル・データベースには test という名前が付けられます。

```
dbsrv9 -c 3Mb "c:¥Program Files¥Sybase¥SQL Anywhere
9¥asademo.db" -n "test"
```

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、サーバ SERV とデータベース DATA が指定されます。

```
dbsrv9 -n SERV asademo.db -n DATA
```

詳細については、「-n サーバ・オプション」200 ページを参照してください。

-r データベース・オプション

機能

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。このオプションの内容は指定する位置によって異なります。

構文

{ **dbsrv9** | **dbeng9** } **-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

説明

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。

変更できないデータベース・ファイルの例として、CD-ROM デバイスで配布されているデータベースや圧縮されているデータベースなどが挙げられます。ライト・ファイルを作成してデータベース・ファイル以外のデータベースに変更を加えるか、読み込み専用モードで実行することはできます。

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。

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。

```
dbeng9 -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。

```
dbeng9 database1.db -r database2.db
```

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

この章の内容

この章では、クライアント・アプリケーションからデータベースへの接続を確立し、記述するパラメータについて説明します。

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

  詳細については、「接続パラメータ・リストのアセンブル」90 ページを参照してください。

- ODBC データ・ソース

  詳細については、「ODBC データ・ソースの使用」70 ページを参照してください。

- [Adaptive Server Anywhere 接続] ダイアログ

  詳細については、「Adaptive Server Anywhere ユーティリティからの接続」68 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 [Adaptive Server Anywhere 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

- 接続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**　Adaptive Server 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、Adaptive Server 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**　Adaptive Server 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、asademo という名前のパーソナル・サーバがデータベース *asademo* を実行しているとします (これはコマンド `dbeng9 asademo.db` で実行できます)。

  次の文字列は、「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=asademo;dbn=asademo;uid=db;pwd=sql"
  ```

  ただし、次のコマンドは、「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_engine;uid=dba;pwd=sql"
  ```

  詳細については、「Ping ユーティリティ」730 ページを参照してください。

AppInfo 接続パラメータ [APP]

機能　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*

説明　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・マシンの 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することができます。

```
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、または ADO.NET (ODBC は iAnywhere JDBC ドライバの応答)
- **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・マシンのホスト名
- **IP** クライアント・マシンの IP アドレス (UNIX と NetWare のみ)
- **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、使用している接続プロトコルのバージョン (例、9.0.00.3642)

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」52 ページ
- 「接続パラメータのヒント」84 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=C:¥Program
  Files¥Sybase¥SQL Anywhere 9¥asademo.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)。

  ```
  HOST=machine-name;OS=Windows 2000 Build 2195 Service
  Pack 3;PID=0x724;THREAD=0x6bc;EXE=C:¥Program
  Files¥Sybase¥SQL Anywhere
  9¥win32¥dbisqlg.exe;VERSION=9.0.00.3642;API=ODBC;T
  IMEZONEADJUSTMENT=-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=C:¥Program
  Files¥Sybase¥SQL Anywhere 9¥asademo.db;app=ISQL
  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)。

  ```
  HOST=machine-name;OS=Windows 2000 Build 2195 Service
  Pack 3;PID=0x8d0;THREAD=0xd74;EXE=C:¥Program
  Files¥Sybase¥SQL Anywhere
  9¥win32¥dbisqlg.exe;VERSION=9.0.00.3642;API=ODBC;T
  IMEZONEADJUSTMENT=-300;APPINFO=ISQL connection
  ```

AutoStart 接続パラメータ [ASTART]

機能

接続が検出できない場合にローカル・データベース・サーバが起動されることを回避します。

使用法

特に制限なし

値の範囲

YES、**NO**

デフォルト

YES

説明

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・サーバは同一のマシンで起動されます。接続文字列の AutoStart (ASTART) 接続パラメータを **NO** または **OFF** 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。

参照

- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 ページ

AutoStop 接続パラメータ [ASTOP]

機能

オープン接続がなくなった場合に、すぐにデータベースが停止されることを回避します。

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

説明

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」52 ページ
- 「接続パラメータのヒント」84 ページ

CharSet 接続パラメータ [CS]

機能

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

値の範囲

文字列

デフォルト

ローカル文字セット

指定する方法の詳細については、「ロケール情報の確認」460 ページを参照してください。

説明

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。

有効な文字セット値のリストについては、「文字セット・ラベル」480 ページを参照してください。

参照

- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 ページ

CommBufferSize 接続パラメータ [CBSIZE]

機能

通信パケットの最大サイズをバイトで設定します。

使用法

特に制限なし

値の範囲

整数

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。

説明

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。CommBufferSize の最小値は 300 バイトで、最大値は 16,000 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」52 ページ
- 「接続パラメータのヒント」84 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]

機能

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。

値の範囲

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。

説明

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マシン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、同じマシン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。

`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。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)** 同一マシン通信の共有メモリ・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含まれる場合は、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。
- **NamedPipes (NP)** C2 セキュリティのために、Windows NT/2000/XP クライアントから -sc オプションを使って起動された同じマシン上のデータベース・サーバに接続します。
- **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。

パラメータ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」276 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」276 ページ
- 「クライアント／サーバ通信」113 ページ

- 「-x サーバ・オプション」218 ページ
- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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 ポートを起動し、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 ポートを起動し、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 ポートを起動し、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 ポートは起動*されません*。

```
CommLinks=spx,tcpip(HOST=kangaroo)
```

Compress 接続パラメータ [COMP]

機能

通信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、Adaptive Server Anywhere 通信圧縮はサポートされません。

値の範囲

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 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。

説明

Adaptive Server 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。

Compress (COMP) パラメータの指定可能な値は、大文字と小文字を区別しません。次の値が含まれます。

- **YES** この接続の通信圧縮をオンに設定します。
- **NO** この接続の通信圧縮をオフに設定します。

時間を節約し、期待どおりの結果が得られるように、特定のアプリケーションを使用して特定のネットワークのパフォーマンスを分析してから、運用環境で通信圧縮機能を使用してください。

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、同一マシン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」203 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」125 ページ
- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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]

機能

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。

値の範囲

圧縮するパケットの最小バイト・サイズを示す*整数*。80 未満の値はおすすめしません。

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。

説明

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、Adaptive Server 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」204 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」125 ページ
- 「接続パラメータの働き」52 ページ

- 「接続パラメータのヒント」84 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]

機能

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。

値の範囲

文字列

デフォルト

接続名なし

説明

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」52 ページ
- 「接続パラメータのヒント」84 ページ

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]

機能

まだ実行されていないデータベースを起動する場合に使用します。DatabaseFile 接続パラメータは、ロードして接続するデータベース・ファイルを示します。

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

値の範囲　*文字列*

デフォルト　デフォルト設定なし

説明

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、Adaptive Server 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。

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、「データベース・サーバ」154 ページを参照してください。

警告
データベース・ファイルは、データベース・サーバと同じマシン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

- 「DatabaseName 接続パラメータ [DBN]」251 ページ
- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 ページ

例

- 次の例では、DatabaseFile (DBF) 接続パラメータが、ディレクトリ *c:¥Program Files¥Sybase¥SQL Anywhere 9* にインストールされているサンプル・データベース *asademo.db* をロードして接続します。

  ```
  DBF=c:¥Program Files¥Sybase¥SQL
  Anywhere 9¥asademo.db
  ```

- 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。

  ```
  dbeng9 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]

機能 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

値の範囲 *文字列*

デフォルト なし

説明

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。

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。

詳細については、『SQL Anywhere Studio セキュリティ・ガイド』>「Adaptive Server Anywhere トランスポート・レイヤ・セキュリティ」を参照してください。

参照

- 「-ek データベース・オプション」229 ページ
- 「-ep サーバ・オプション」184 ページ
- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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=C:¥Prog
ram Files¥Sybase¥SQL Anywhere 9¥asademo.db"
```

DatabaseName 接続パラメータ [DBN]

機能

すでに実行されているデータベースに接続する場合に使用します。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

値の範囲

文字列

デフォルト

デフォルト設定なし

説明

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」52 ページ
- 「接続パラメータのヒント」84 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。

```
dbeng9 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]

機能　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 (スイッチ) 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。

値の範囲　*文字列*

デフォルト　*オプションなし*

説明　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データベース・オプション (スイッチ) のみを指定できます。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・オプションの詳細については、「データベース・オプション」229 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

- 「データベース・サーバ」154 ページ
- 「StartLine 接続パラメータ [START]」273 ページ
- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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 *asademo.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=c:¥Program
Files¥Sybase¥SQL Anywhere
9¥asademo.db;dbn=my_db;dbs=-r"
```

DataSourceName 接続パラメータ [DSN]

機能
ODBC ドライバ・マネージャまたは Embedded SQL ライブラリに対して、*odbc.ini* ファイルまたはレジストリ内での 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

値の範囲
文字列

デフォルト
デフォルト・データ・ソース名なし

説明
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。

Adaptive Server 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]」262 ページ
- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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=Dynamo Demo
```

DisableMultiRowFetch 接続パラメータ [DMRF]

機能

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

値の範囲

YES、NO

デフォルト

NO

説明

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 **ON** に設定すると、この動作を無効にできます。

詳細については、『ASA SQL ユーザーズ・ガイド』>「プロシージャとトリガでのカーソルの使用」を参照してください。

DisableMultiRowFetch (DMRF) 接続パラメータを **ON** に設定するのと、**PREFETCH** データベース・オプションを **OFF** に設定するのは同じ効果があります。

詳細については、『ASA プログラミング・ガイド』>「ローのプリフェッチ」を参照してください。

参照

- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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]

機能　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

値の範囲　*文字列*

デフォルト　なし

説明　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。

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」52 ページ
- 「接続パラメータのヒント」84 ページ

Encryption 接続パラメータ [ENC]

機能　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　**ECC_TLS** (Certicom)、**RSA_TLS**、**RSA_TLS**、**RSA_TLS_FIPS** の場合は TCP/IP のみ

NONE または **SIMPLE** の場合は特に制限なし

値の範囲　*文字列*

デフォルト　**NONE**

Encryption 値が設定されていない場合、暗号化はサーバ側の設定によって制御されます。

サーバの暗号化設定については、「-ec サーバ・オプション」180 ページを参照してください。

説明

このパラメータは、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・サーバ間の通信を安全化する場合に使用します。

詳細については、『SQL Anywhere Studio セキュリティ・ガイド』>「Adaptive Server Anywhere トランスポート・レイヤ・セキュリティ」を参照してください。

個別にライセンスを取得可能なオプションが必要

トランスポート・レイヤ・セキュリティを使用するには、個別にライセンスを取得可能な SQL Anywhere Studio セキュリティ・オプションを入手する必要があります。このセキュリティ・オプションは、輸出規制対象品目です。

このコンポーネントを注文するには、『SQL Anywhere Studio の紹介』>「別途ライセンスが入手可能なコンポーネント」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。この値は、Adaptive Server Anywhere の以前のバージョンの NO 設定に相当します。
- **simple**　すべてのプラットフォームと Adaptive Server 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。この値は、Adaptive Server Anywhere の以前のバージョンの YES 設定に相当します。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。
- **ECC_TLS**　楕円曲線に基づく Certicom 暗号化テクノロジで暗号化された通信パケットを受け入れます。下位互換性を保つために、ecc_tls を **CERTICOM** と指定することもできます。このタイ

プの暗号化を使用するには、Solaris、Linux、NetWare、Mac OS X、またはこの暗号化をサポートしている Windows オペレーティング システム (WindowsCE 以外) で、サーバとクライアントの両方を実行してください。また、接続には TCP/IP ポートを使用します。Solaris、Linux、および Mac OS X 以外の UNIX プラットフォームでは、クライアントまたはサーバの ECC_TLS パラメータは認識されません。クライアントは、サーバのパブリック証明書のフィールド値の検証に、次の引数を使用できます。

- **trusted_certificates** クライアントがサーバを認証するときに使用する証明書ファイルを指定します。これは唯一必須のパラメータです。
- **certificate_company** 証明書の組織フィールドの値を指定します。
- **certificate_unit** 証明書の組織単位フィールドの値を指定します。
- **certificate_name** 証明書の通称を指定します。

  サーバ認証での証明書のフィールドの検証については、『SQL Anywhere Studio セキュリティ・ガイド』> 「証明書フィールドの確認」を参照してください。

  デジタル証明書の使用については、『SQL Anywhere Studio セキュリティ・ガイド』> 「デジタル証明書の作成」を参照してください。

- **RSA_TLS** RSA 暗号化テクノロジで暗号化された通信パケットを受け入れます。このタイプの暗号化を使用するには、Solaris、Linux、NetWare、Mac OS X、またはこの暗号化をサポートしている Windows オペレーティング システム (WindowsCE 以外) で、サーバとクライアントの両方を実行してください。また、接続には TCP/IP ポートを使用します。Solaris、Linux、および Mac OS X 以外の UNIX プラットフォームでは、クライアントまたはサーバの RSA_TLS パラメータは認識されません。クライアントは、サーバのパブリック証明書のフィールド値の検証に、次の引数を使用できます。

- **trusted_certificates**　クライアントがサーバを認証するときに使用する証明書ファイルを指定します。これは唯一必須のパラメータです。
- **certificate_company**　証明書の組織フィールドの値を指定します。
- **certificate_unit**　証明書の組織単位フィールドの値を指定します。
- **certificate_name**　証明書の通称を指定します。

  サーバ認証での証明書のフィールドの検証については、『SQL Anywhere Studio セキュリティ・ガイド』> 「証明書フィールドの確認」を参照してください。

  デジタル証明書の使用については、『SQL Anywhere Studio セキュリティ・ガイド』> 「デジタル証明書の作成」を参照してください。

- **RSA_TLS_FIPS**　FIPS 承認の RSA 暗号化テクノロジで暗号化された通信パケットを受け入れます。RSA_TLS_FIPS は、別の承認済みライブラリを使用しますが、Adaptive Server Anywhere 9.0.2 以降で RSA_TLS を指定するサーバと互換性があります。このタイプの暗号化を使用するには、クライアントとサーバの両方が、サポートされている 32 ビットの Windows オペレーティング・システムで実行されている必要があります。また、TCP/IP ポートで接続する必要があります。クライアントは、サーバのパブリック証明書のフィールド値の検証に、次の引数を使用できます。

  - **trusted_certificates**　クライアントがサーバを認証するときに使用する証明書ファイルを指定します。これは唯一必須のパラメータです。
  - **certificate_company**　証明書の組織フィールドの値を指定します。
  - **certificate_unit**　証明書の組織単位フィールドの値を指定します。
  - **certificate_name**　証明書の通称を指定します。

サーバ認証での証明書のフィールドの検証については、『SQL Anywhere Studio セキュリティ・ガイド』>「証明書フィールドの確認」を参照してください。

デジタル証明書の使用については、『SQL Anywhere Studio セキュリティ・ガイド』>「デジタル証明書の作成」を参照してください。

FIPS 承認の RSA 暗号化を使用する場合は、RSA 暗号化を使用して証明書を生成してください。

証明書の詳細については、『SQL Anywhere Studio セキュリティ・ガイド』>「デジタル証明書の作成」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。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 システム関数の使用方法については、『ASA SQL リファレンス・マニュアル』>「CONNECTION_PROPERTY 関数 [システム]」を参照してください。

参照

- 『SQL Anywhere Studio セキュリティ・ガイド』>「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」
- 「-ec サーバ・オプション」180 ページ
- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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 リンクを通じてデータベース・サーバ myeng に接続します。

```
"ENG=myeng;LINKS=tcpip;Encryption=ECC_TLS
(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 リンクを通じてデータベース・サーバ myeng に接続します。

```
"ENG=myeng;LINKS=tcpip;Encryption=RSA_TLS
(trusted_certificates=sample.crt)"
```

次の接続文字列フラグメントは、単純暗号化を使用して、TCP/IP リンクを通じてデータベース・サーバ myeng に接続します。

```
"ENG=myeng;LINKS=tcpip;Encryption=SIMPLE"
```

EngineName 接続パラメータ [ENG]

機能　ServerName の同義語。接続する実行中のデータベース・サーバ名です。

使用法　ネットワーク・サーバか実行中のパーソナル・サーバ

値の範囲　*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

説明　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・サーバが実行中のときか、ネットワーク・サーバに接続するときのみ、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・マシンの文字セットに従って解釈されます。サーバ名には、マルチバイト文字を使用しないでください。

この名前は、有効な識別子である必要があります。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
UNIX 共有メモリ	31 バイト
UNIX 以外の共有メモリ	40 バイト

プロトコル	トランケーションの長さ
TCP/IP	40 バイト
SPX	32 バイト
名前付きパイプ	8 バイト

注意
配備されたアプリケーションの場合は、接続文字列に 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、マシンで複数の Adaptive Server Anywhere データベース・サーバが実行されている場合、アプリケーションが正しいサーバに接続することが保証され、タイミング依存の接続エラーを防ぐことができます。

参照

- 『ASA SQL リファレンス・マニュアル』> 「識別子」
- 「-n サーバ・オプション」200 ページ
- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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]

機能
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

値の範囲
文字列

デフォルト
デフォルト名なし

説明

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、接続情報を各マシン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]」254 ページ
- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 ページ

ForceStart 接続パラメータ [FORCESTART]

機能

サーバに接続せずに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

値の範囲

YES、NO

デフォルト

NO

説明

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

- 『ASA プログラミング・ガイド』> 「db_start_engine 関数」
- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 ページ

Idle 接続パラメータ [IDLE]

機能

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、Adaptive Server 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 240

説明 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・タイムアウト値に優先されます。

参照

- 「-ti サーバ・オプション」210 ページ
- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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]

機能 統合化ログイン機能を使用します。

使用法 特に制限なし

値の範囲 **YES**、**NO**

デフォルト **NO**

説明 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 Mixed 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 データーベース・オプションを Mixed か 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]」862 ページ
- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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
```

Language 接続パラメータ [LANG]

機能

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、言語をドイツ語に設定します。

デフォルト

ASLANG 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

説明

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、ASLANG 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。

言語コードのリストについては、「ロケール言語の知識」434 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。Adaptive Server Anywhere の各種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」52 ページ
- 「接続パラメータのヒント」84 ページ

LazyClose 接続パラメータ [LCLOSE]

機能

このオプションを有効にすると、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

値の範囲

YES、NO

デフォルト

NO

説明

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」52 ページ
- 「接続パラメータのヒント」84 ページ

LivenessTimeout 接続パラメータ [LTO]

機能

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

120

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。

説明

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。

参照

- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 ページ

例

- 次の例では、LivenessTimeout 値を 10 分に設定します。

```
LTO=600
```

Logfile 接続パラメータ [LOG]

機能

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

値の範囲

文字列

デフォルト

ログ・ファイルなし

説明

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ある場合、パスはクライアント・アプリケーションの現在の作業ディレクトリに対する相対パスです。

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。

```
Fri Oct 27 2000 11:45
 Application information:
 "HOST=NAME-PC;OS=Windows NT 4.0 (Service Pack
5);PID=0x14b;THREAD=0x148;EXE=C:¥ASA90¥WIN32¥DBPING.EXE
;VERSION=9.0.0.1271;API=DBLIB;TIMEZONEADJUSTMENT=-300"
 Attempting to connect using:

UID=dba;PWD=***;ENG=name;DBG=YES;LOG=c:¥temp¥cli.out;LI
NKS=shmem,tcpip
 Attempting to connect to a running server...
 Trying to start SharedMemory link ...
     SharedMemory link started successfully
 Attempting SharedMemory connection (no asasrv.ini
cached address)
 Failed to connect over SharedMemory
 Trying to start TCPIP link ...
 Loading wsock32.dll
 Loading ws2_32.dll
 TCP using Winsock version 2.0
 My IP address is 172.31.142.196
 My IP address is 127.0.0.1
     TCPIP link started successfully
```

```
 Attempting TCPIP connection (address
172.31.143.196:2638 found in asasrv.ini cache)
 Trying to find server at cached address
172.31.143.196:2638 without broadcasting
     Server not found (no reply received)
 Looking for server with name NAME
 I am in a class B network
 Sending broadcast to find server
 Using broadcast address of: 172.31.255.255:2638
 I am in a class A network
 Sending broadcast to find server
 Using broadcast address of: 127.255.255.255:2638
 Found database server at address 172.31.142.196:2638
 Found database server NAME on TCPIP link
 Connected to server over TCPIP at address
172.31.142.196:2638
 Writing server address 172.31.142.196:2638 to
asasrv.ini cache
    Liveness timeout 120, liveness retransmit period 30
 Connected to the server, attempting to connect to a
running database...
 Connected to database successfully
```

参照

- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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、ASA 9.0 サンプル・データ・ソースに接続された Interactive SQL を起動します。

```
dbisql -c "DSN=ASA 9.0 Sample;LOG=d:¥logs¥test.txt"
```

Password 接続パラメータ [PWD]

機能

接続時のパスワードを指定します。

使用法

特に制限なし

値の範囲

文字列

デフォルト

パスワードの指定なし

説明

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。パスワードの長さは最大で 255 バイトです。

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 Adaptive Server Anywhere ODBC 設定ツールのいずれも、暗号化したパラメータを使用します。

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]」256 ページ
- 『ASA SQL リファレンス・マニュアル』>「GRANT 文」
- 『ASA SQL ユーザーズ・ガイド』>「大文字と小文字の区別」
- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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]

機能

ローをバッファするためのメモリの最大容量を、キロバイト単位で設定します。

使用法

特に制限なし

値の範囲

整数

デフォルト

16 (Windows CE) **64** (その他のプラットフォーム)

説明

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。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。

詳細については、「PrefetchRows 接続パラメータ [PROWS]」272 ページを参照してください。

参照

- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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;Debug=YES;Logfile=c:¥client.txt
```

- 次の文字列を使用して、メモリ制限を 256 K に増やすことができます。

```
...PrefetchRows=100;PrefetchBuffer=256
```

PreFetchOnOpen 接続パラメータ

使用法　ODBC

値の範囲　**YES、NO**

デフォルト　**NO**

説明

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]

機能

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

値の範囲

整数

デフォルト

10

説明

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マシンで動作しているか、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。

詳細については、「PrefetchBuffer 接続パラメータ [PBUF]」270 ページを参照してください。

参照

- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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;...
```

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。

詳細については、「EngineName 接続パラメータ [ENG]」261 ページ を参照してください。

StartLine 接続パラメータ [START]

機能

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

値の範囲 *文字列*

デフォルト StartLine パラメータなし

説明 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 (START)** 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。

使用できるコマンド・ライン・オプションの詳細については、「データベース・サーバ」154 ページを参照してください。

参照

- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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=dbeng9 -c 8M;dbf=asademo.db
```

Unconditional 接続パラメータ [UNC]

機能 サーバへの接続があるときでも、dbstop を使用してサーバを停止します。

使用法 特に制限なし

値の範囲 **YES**、**NO**

デフォルト **NO**

説明 dbstop ユーティリティはデータベース・サーバを停止します。接続文字列で `UNC=YES` を指定すると、サーバ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サーバが停止されます。

参照

- 「dbstop コマンド・ライン・ユーティリティを使用したデータベース・サーバの停止」749 ページ

- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 ページ

例

- 次のコマンド・ラインでは、無条件にサーバを停止します。

```
dbstop -c "UID=DBA;PWD=SQL;ENG=server-
name;UNC=YES"
```

Userid 接続パラメータ [UID]

機能　データベースにログオンするユーザ ID です。

使用法　特に制限なし

値の範囲　*文字列*

デフォルト　*なし*

説明　データベースに接続するときは、必ずユーザ ID を指定してください。

参照

- 「接続パラメータの働き」52 ページ
- 「接続パラメータのヒント」84 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。

```
dbsrv9 -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。

```
dbsrv9 -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は、二重引用符が必要です。

ブール・パラメータは、YES、ON、TRUE、1 のいずれかによってオンになり、NO、OFF、FALSE、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]

TCP/IP	HTTP と HTTPS	SPX
BroadcastListener [BLISTENER]	Certificate_Password	DLL
ClientPort [CPORT]	DatabaseName [DBN]	DoBroadcast [DOBROAD]
DLL	LocalOnly [LOCAL]	ExtendedName [ENAME]
DoBroadcast [DOBROAD]	LogFile [LOG]	Host [IP]
Host [IP]	LogMaxSize [LSize]	RegisterBindery [REGBIN]
LocalOnly [LOCAL]	LogOptions [LOpt]	SearchBindery [BINSEARCH]
LDAP [LDAP]	LogFormat [LF]	Timeout [TO]
MyIP [ME]	MaxConnections [MaxConn]	
ReceiveBufferSize [RCVBUFSZ]	MaxRequestSize [MaxSize]	
SendBufferSize [SNDBUFSZ]	MyIP [ME]	
ServerPort [PORT]	ServerPort [PORT]	
TDS	Timeout [TO]	
Timeout [TO]		
VerifyServerName [VERIFY]		

Broadcast プロトコル・オプション [BCAST]

使用法　TCP/IP

値の範囲　*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

説明 BROADCAST は、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。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。

BroadcastListener プロトコル・オプション [BLISTENER]

使用法 SPX、TCP/IP、サーバ側

値の範囲 **YES**、**NO**

デフォルト **YES**

説明 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」209 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 接続を必要とするサーバを起動します。

```
dbsrv9 -x tcpip(BroadcastListener=NO),spx ...
```

Certificate プロトコル・オプション

使用法 HTTPS

値の範囲 *文字列*

デフォルト　デフォルトの証明書名なし

説明　このオプションにより、暗号化された証明書の名前を指定できます。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。

```
dbsrv9 -xs
https(Certificate=cert.file;Certificate_Password=s
ecret) ...
```

Certificate_Password プロトコル・オプション

使用法　HTTPS

値の範囲　*文字列*

デフォルト　デフォルトの証明書名なし

説明　このオプションにより、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できます。

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。

```
dbsrv9 -xs
https(Certificate=cert.file;Certificate_Password=s
ecret) ...
```

ClientPort プロトコル・オプション [CPORT]

使用法　TCP/IP (クライアント側のみ)

値の範囲　*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。

説明

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することができます。

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]」284 ページ
- 「DoBroadcast プロトコル・オプション [DOBROAD]」282 ページ
- 「ServerPort プロトコル・オプション [PORT]」294 ページ
- 「ファイアウォール経由の接続」116 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 ～ 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);
 gServerName=my_server
```

DatabaseName プロトコル・オプション [DBN]

使用法　HTTP、HTTPS

値の範囲　**AUTO**、**REQUIRED**、*データベース名*

デフォルト　**AUTO**

説明　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。

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。

```
dbsrv9 -xs http(dbn=web) asademo.db web.db
```

DLL プロトコル・オプション

使用法　TCP/IP、SPX (Windows 95/98/Me、Windows NT/2000/XP)

値の範囲 *文字列*

デフォルト

- Windows CE を含むすべての Windows プラットフォームのデータベース・サーバでは、デフォルトは **ws2_32.dll** (Winsock 2.0) です。
- Windows NT/2000/XP のクライアントでは、デフォルトは **ws2_32.dll** (Winsock 2.0) です。
- Windows 95/98/Me および Windows CE のクライアントでは、デフォルトは **wsock32.dll** (Winsock 1.1) です。

説明 ネットワーク・インタフェースから呼び出される機能のうちで、未試行の TCP/IP プロトコル・スタックは、デフォルトのプロトコル・スタックと異なる DLL によってサポートされます。クライアントまたはサーバは、要求された機能を名前付き DLL で検索します。

Winsock 2.0 は、すべての Windows プラットフォーム上のデータベース・サーバで必要です。

例

- Windows 98 での次のコマンドは、Winsock 98 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]

使用法 TCP/IP、SPX

値の範囲 **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)

説明 **クライアントでの使用法** `DoBroadcast=ALL` (以前の `DoBroadcast=YES`) に設定した場合、ブロードキャストを実行して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 (以前の DoBroadcast=NO) 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 (*asasrv.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、サーバ名を検証しないように VerifyServerName (VERIFY) プロトコル・オプションを **NO** に設定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、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することができます。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)
asademo
```

ExtendedName プロトコル・オプション [ENAME]

使用法　SPX (Windows 95/98/Me または Windows NT/2000/XP 以外のプラットフォーム)

値の範囲 **YES**、**NO**

デフォルト **NO**

説明 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。

```
¥ / : ; , * ? + -
```

asademo-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 **-** を削除して **asademo1** 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 asademo-1 を起動します。

```
load dbsrv9.nlm -x spx(ExtendedName=YES) asademo-1
```

Host プロトコル・オプション [IP]

使用法 TCP/IP、SPX (すべてのプラットフォーム) サーバおよびクライアント側

値の範囲 *文字列*

デフォルト 追加のマシンなし

説明 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マシンを指定します。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、HOST 値 *hostname* またはピリオドで区切った 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。

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、サーバの起動時にデータベース・サーバ・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、複数のマシンを検索できます。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。

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、**HOST** を使用します。

参照

- 「ClientPort プロトコル・オプション [CPORT]」279 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) というマシンを検索し、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 というマシンを検索し、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。最初に応答があっ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)
```

LDAP プロトコル・オプション [LDAP]

使用法 TCP/IP

値の範囲 **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 *asaldap.ini* です。

説明

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 (および検出ユーティリティ [dblocate]) が LDAP サーバにクエリを実行することができます。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することができます。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、指定したファイルを設定ファイルとして使用します。**LDAP**=*YES* と指定すると、LDAP のサポートがオンになり、*asaldap.ini* を設定ファイルとして使用します。

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 *asaldap.ini* ファイルの内容を非表示にすることができます。

詳細については、「.ini ファイルの内容の非表示」679 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」119 ページ

LocalOnly プロトコル・オプション [LOCAL]

使用法　TCP/IP、HTTP、HTTPS

値の範囲　**YES**、**NO**

デフォルト　**NO**

説明

LocalOnly (LOCAL) プロトコル・オプションは、クライアントがローカル・マシン上のサーバ (存在する場合) のみに接続できるようにします。サーバ名が一致するサーバが、ローカル・マシン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` では、サーバから他のマシン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・マシンへの接続に限定させることができます。リモート・マシン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]」277 ページ

LogFile プロトコル・オプション [LOG]

使用法　HTTP、HTTPS

値の範囲　*ファイル名*

デフォルト　なし

説明 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]」288 ページ
- 「LogMaxSize プロトコル・オプション [LSIZE]」289 ページ
- 「LogOptions プロトコル・オプション [LOPT]」290 ページ

LogFormat プロトコル・オプション [LF]

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**

説明 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]」287 ページ
- 「LogMaxSize プロトコル・オプション [LSIZE]」289 ページ
- 「LogOptions プロトコル・オプション [LOPT]」290 ページ

LogMaxSize プロトコル・オプション [LSIZE]

使用法　HTTP、HTTPS

値の範囲　*サイズ*

デフォルト　**0**

説明　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。

参照

- 「LogFile プロトコル・オプション [LOG]」287 ページ
- 「LogFormat プロトコル・オプション [LF]」288 ページ
- 「LogOptions プロトコル・オプション [LOPT]」290 ページ

LogOptions プロトコル・オプション [LOPT]

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**

説明 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]」287 ページ
- 「LogFormat プロトコル・オプション [LF]」288 ページ
- 「LogMaxSize プロトコル・オプション [LSIZE]」289 ページ

MaxConnections プロトコル・オプション [MAXCONN]

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) またはライセンスされている接続数 (ネットワーク・サーバ)

説明　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]」291 ページ

MaxRequestSize プロトコル・オプション [MAXSIZE]

使用法　HTTP、HTTPS

値の範囲 *サイズ* (バイト)

デフォルト 100 KB

説明 サーバで許可される最大要求サイズ。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]」291 ページ

MyIP プロトコル・オプション [ME]

使用法 TCP/IP、HTTP、HTTPS

値の範囲 *文字列*

説明 MyIP (ME) プロトコル・オプションは、複数のネットワーク・アダプタを持つマシン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・カードを持つマシンや、リモート・アクセス (RAS) ソフトウェアとネットワーク・カードを持つマシンなど、この操作により大きな負荷がかかるマシン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。

このオプションは、Windows 95/98/Me または Windows NT/2000/XP で、複数の IP アドレスを持つマシンに対して何度でも使用できます。

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。

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。

```
dbsrv9 -x tcpip(MyIP=192.75.209.12,192.75.209.32)
"c:¥Program Files¥Sybase¥SQL Anywhere
9¥asademo.db"
```

- 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]

使用法　TCP/IP

値の範囲　*整数* (バイト)

デフォルト　マシンによって異なります

説明　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。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。

RegisterBindery プロトコル・オプション [REGBIN]

使用法　SPX

サーバ側のみ

値の範囲　**YES**、**NO**

デフォルト　**YES**

説明 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]

使用法 SPX

値の範囲 **YES**、**NO**

デフォルト **YES**

説明 `SEARCHBINDERY=NO` の場合、NetWare バインダリではデータベース・サーバは検索されません。

SendBufferSize プロトコル・オプション [SNDBUFSZ]

使用法 TCP/IP

値の範囲 *整数* (バイト)

デフォルト マシンによって異なります

説明 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。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]

使用法 TCP/IP、HTTP、HTTPS

値の範囲 *整数*

デフォルト

TCP/IP のデフォルト値は **2638** です。HTTP のデフォルト値は **80** です。HTTPS のデフォルト値は **443** です。

説明

Internet Assigned Numbers Authority は、Adaptive Server 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。

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、HP-UX、および Tru64 では、サーバは UDP ポート 2638 で受信*しません*が、その他のプラットフォーム上のサーバは UDP ポート 2638 で受

信します。Mac OS X、HP-UX、および Tru64 プラットフォームでは、TCP/IP ポート 2638 で第 2 のデータベース・サーバが起動されたときのために、UDP ポート 2638 は使用可能な状態にしておきます。

Mac OS X、HP-UX、Tru64 の違い

Mac OS X、HP-UX、および Tru64 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 `dbsrv9 -n MyASAServer asademo.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=MyASAServer;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 `dbsrv9 -n SecondASAServer -x tcpip(PORT=7777) asademo.db` を使って Mac OS X、HP-UX、または Tru64 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=SecondASAServer;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、HP-UX、または Tru64 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、HP-UX、および Tru64 では、Adaptive Server 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。

   ```
   dbsrv9 -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。

   ```
   dbsrv9 -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、HP-UX、および Tru64 では、これは失敗します)。

3. すでにマシン上の別の 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。

   ```
   dbsrv9 -xs http(port=8080) -n server3 web.db
   ```

TDS プロトコル・オプション

使用法　TCP/IP、名前付きパイプ

サーバ側のみ

値の範囲　**YES**、**NO**

デフォルト　**YES**

説明　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れらのポート・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。

  ```
  dbsrv9 -x tcpip(TDS=NO) ...
  ```

Timeout プロトコル・オプション [TO]

使用法 TCP/IP、SPX、HTTP、HTTPS

値の範囲 *整数* (秒)

デフォルト **5** (TCP/IP の場合)、**30** (HTTP と HTTPS の場合)

説明 TIMEOUT は、通信確立時に、応答を待つ時間を秒単位で設定します。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。

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]

使用法 TCP/IP

クライアント側のみ

値の範囲 **YES**、**NO**

デフォルト **YES**

説明

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、Adaptive Server Anywhere クライアントから Adaptive Server 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。

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]」282 ページ

第 7 章

Web サービスの使用

この章の内容

Adaptive Server Anywhere には、Web サービスを提供したり、その他の Adapter Server Anywhere データベースの Web サービスやインターネットを介して使用できる標準の Web サービスにアクセスするための組み込みの HTTP サーバが含まれています。SOAP はこの目的に使用される標準ですが、Adaptive Server Anywhere の組み込みの HTTP サーバでは、クライアント・アプリケーションからの標準の HTTP 要求や HTTPS 要求も処理できます。この章では、両方のタイプの Web サービスの作成方法と使用方法について説明します。

Web サービスについて

*Web サービス*という用語は、さまざまな意味で使用されています。通常は、マシン間のデータの転送と相互運用性を可能にするソフトウェアを指します。本質的に、Web サービスはビジネス論理のセグメントをインターネットを介して使用できるようにします。*Simple Object Access Protocol* (SOAP) は、このアプリケーションで使用する標準プロトコルで、HTTP を介して XML と通信できるアプリケーションならどれにでも組み込むことができます。

SOAP という略語は、Java または Visual Studio .NET で書かれたアプリケーションなどからアクセスできる柔軟なメッセージ交換プロトコルを指します。SOAP メッセージは、サーバが提供するサービスを定義します。実際のデータ転送は、関連情報を効果的にエンコードするように構造化された XML ドキュメントを交換するために、通常 HTTP を使用して行われます。SOAP 通信に参加するクライアントまたはサーバなどのアプリケーションはすべて SOAP ノードまたは SOAP 終了ポイントと呼ばれます。このようなアプリケーションは、SOAP メッセージを送信、受信、処理できます。SOAP ノードは、Adaptive Server Anywhere を使用して作成できます。

Web サービスと SQL Anywhere

SQL Anywhere のコンテキストでは、Web サービスという用語は Adaptive Server Anywhere に標準の SOAP 要求を受信し処理する機能があることを意味しています。Adaptive Server Anywhere では、Web サービスはクライアント・アプリケーションに対して、JDBC や ODBC などの従来のインタフェースの代わりに使用できるものを提供します。Web サービスへは、さまざまな言語で書かれ、さまざまなプラットフォームで実行されているクライアント・アプリケーションからアクセスできます。Perl や Python などの一般的なスクリプト言語でも Web サービスにアクセスできます。Web サービスは、CREATE SERVICE 文を使用してデータベースに作成できます。

Adaptive Server Anywhere は、SOAP または HTTP クライアントとしても機能し、データベース内で実行中のアプリケーションがインターネットで使用できる標準の Web サービスまたは Adaptive Server Anywhere データベースで提供されている Web サービスにアクセスできるようにします。このクライアント機能は、ストアド関数またはストアド・プロシージャを使ってアクセスされます。

さらに、この用語は組み込みの Web サーバを使用してクライアントからの HTTP 要求を処理するアプリケーションを指します。通常これらのアプリケーションは、従来のデータベースに基づく Web アプリケーションのように機能しますが、よりコンパクトで、データとアプリケーション全体がデータベース内に存在できるため、書き込みが簡単です。このアプリケーションでは、Web サービスは通常 HTML フォーマットのドキュメントを返します。GET、HEAD、POST メソッドをサポートしています。

データベース内の Web サービスの集合が、使用可能な URL を定義します。各サービスには Web ページのセットがあります。通常、これらのページの内容はデータベース内に記述および格納されたプロシージャによって生成され、単一文の場合もあれば、オプションでユーザ自身が文を実行することもできます。これらの Web サービスは、HTTP 要求の受信を可能にするスイッチのあるデータベース・サーバを起動すると使用可能になります。

Web サービス要求を処理する HTTP サーバがデータベースに組み込まれているため、パフォーマンスに優れています。Web サービスを使用するアプリケーションは、データベースとデータベース・サーバ以外の追加のコンポーネントが必要ないため、簡単に配備されます。

SOAP 標準の詳細については、www.w3.org/TR/SOAP を参照してください。

クイック・スタート

ここでは、新しいデータベースの作成、HTTP サーバを有効にした Adaptive Server Anywhere データベースの起動、一般的な Web ブラウザを使用したデータベースへのアクセス方法について説明します。

❖ 簡単な XML Web サービスを作成しアクセスするには、次の手順に従います。

1 このデモに使用する新しいデータベースを作成するために、コマンド・プロンプトで次の文を実行します。

```
dbinit web.db
```

2 次の文を実行して、パーソナル Web サーバを起動します。-xs http(port=80) スイッチは、HTTP 要求を受信するようにデータベースに指示します。ポート 80 ですでに Web サーバが実行されている場合、このデモには 8080 などの別のポート番号を使用します。

```
dbeng9 -xs http(port=80) web.db
```

3 Interactive SQL を起動します。ユーザ名 DBA とパスワード SQL を使用して Web データベースに接続します。次の文を実行します。

```
CREATE SERVICE mytables
 TYPE 'XML'
 AUTHORIZATION OFF
 USER DBA
 AS SELECT * FROM SYSTABLE
```

この文は、mytables という Web サービスを作成します。この単純なサービスは SELECT * FROM SYSTABLE 文の結果を返し、出力は自動的に XML フォーマットに変換されます。

4 Web ブラウザを起動します。URL http://localhost:80/web/mytables を検索します。データベース・サーバの起動時に指定したポート番号を使用します。

Web ブラウザは、データベース・サーバによって返された XML ドキュメントの本文を表示します。フォーマット情報は含まれないため、表示されるのはタグや属性を含んだ未加工 XML です。

5　また、一般的なプログラミング言語からも mytable ルーチンにアクセスできます。たとえば、次の短い C# プログラムでは mytable Web サービスを使用します。

```
static void Main(string[] args) {
   XmlTextReader reader = new XmlTextReader( "http:/
/localhost/mytables" );

   while( reader.Read() ) {
     switch( reader.NodeType ) {
     case XmlNodeType.Element:
       if( reader.Name == 'row' ) {

Console.writeline(reader.GetAttribute("table_id");

Console.writeline(reader.GetAttribute("table_name")
;
       }
       break;
     }
   }
```

6　さらに、次のコード例のように Python からも同じ Web サービスにアクセスできます。

```
import xml.sax

 class DocHandler( xml.sax.ContentHandler ):
   def startElement( self, name, attrs ):
     if name == 'row':
       table_id = attrs.getValue( 'table_id' )
       table_name = attrs.getValue( 'table_name' )
       print '%s %s' % ( table_id, table_name )

 parser = xml.sax.makd_parser()
 parser.setContentHandler( DocHandler() )
 parser.parse( 'http://localhost/mytables' )
```

❖ 簡単な HTML Web サービスを設定するには、次の手順に従います。

1 ポート 80 で実行している Web サーバがないことを確認するか、または残りの手順で使用するために別のポート番号を選択します。

2 適切なディレクトリに移動して、web という名前の新しいデータベースを作成します。

```
dbinit web.db
```

3 パーソナル・データベース・サーバを起動して、この新しいデータベースを実行します。-xs オプションが、HTTP を含むすべての通信プロトコルを起動するようにデータベース・サーバに通知します。

```
dbeng9 web.db -xs http(port=80)
```

4 Interactive SQL を起動して、ユーザ ID **DBA** とパスワード **SQL** を使用して新しいデータベースに接続します。

5 Interactive SQL を使用して、次の SQL 文を実行します。これは、SYS.SYSTABLE を HTTP ユーザに公開する HTML サービスを付加するものです。認証がオフになっているので、Web ブラウザからテーブルへのアクセスには許可が不要です。

```
CREATE SERVICE tables TYPE 'HTML'
 AUTHORIZATION OFF USER DBA
 AS SELECT * FROM SYS.SYSTABLE
```

6 Web ブラウザを起動します。URL `http://localhost:80/web/tables` を検索します。データベース・サーバの起動時に指定したポート番号を使用します。

サービスの多くのプロパティは、-xs コマンド・ライン・オプションのパラメータで制御できます。

詳細については、「ネットワーク・プロトコル・オプション」276 ページを参照してください。

適切な -xs オプション・パラメータでデータベース・サーバを起動し、着信要求に応答するための Web サービスを作成してください。これらは、CREATE SERVICE 文を使用して定義します。

詳細については、「Web サービスの作成」308 ページと 『ASA SQL リファレンス・マニュアル』> 「CREATE SERVICE 文」を参照してください。

例

サンプルは、SQL Anywhere Studio インストールの *Samples¥ASA¥HTTP* サブディレクトリにあります。

その他の例は、CodeXchange (http://ianywhere.codexchange.sybase.com/) から入手できます。

Web サービスの作成

データベース内に作成および格納されているサービスが、有効な URL と実行する内容を定義します。単一のデータベースに複数のサービスを定義できます。異なるデータベースにある複数のサービスを、単一の Web サイトの一部として表示するように定義できます。

次の文は、Web サービスの作成、変更、削除を許可します。

- **CREATE SERVICE**
- **ALTER SERVICE**
- **DROP SERVICE**
- **COMMENT ON SERVICE**

CREATE SERVICE 文の一般的な構文は、次のとおりです。

CREATE SERVICE *service-name* **TYPE** *service-type-string* [*attributes*] [**AS** *statement*]

サービス名

サービス名がサービスへのアクセスに使用する URL の一部となるので、URI を構成する文字については柔軟性があります。標準的な英数字に加えて、"-"、"_"、"."、"!"、"*"、"'"、"("、")" が使用できます。

また、サービス名にスラッシュ (/) を含めることができますが、この文字は標準の URL デリミタであり、Adaptive Server Anywhere による URL の解釈方法に影響するため、使用にはいくつかの制限があります。この文字はサービス名の先頭文字としては使えません。また、サービス名に 2 つのスラッシュを連続して使用することはできません。

グループ名にも、DISH サービスのサービス名に適用するものと同じ文字を使用できます。

サービス・タイプ

サポートされるサービス・タイプは次のとおりです。

- **SOAP**　結果セットは SOAP 応答として返されます。データのフォーマットは FORMAT 句によって決定されます。SOAP サービスの要求は有効な SOAP 要求で、簡単な HTTP 要求ではありません。
- **DISH**　DISH サービス (Determine SOAP Handler) は GROUP 句で識別された SOAP サービスのプロキシとして機能し、それらの各 SOAP サービスの WSDL (Web Services Description Language) ファイルを生成します。
- **XML**　結果セットは XML として返されます。結果セットがすでに XML になっている場合、追加のフォーマットは適用されません。まだ XML フォーマットになっていない場合は、自動的に XML フォーマットに変換されます。効果は SELECT 文で FOR XML RAW 句と使用した場合と同様です。
- **HTML**　文またはプロシージャの結果セットは、テーブルを格納する HTML ドキュメントに自動的にフォーマットされます。
- **RAW**　結果セットがこれ以上フォーマットされることなくクライアントに送信されます。要求されたタグをプロシージャ内で明示的に生成することによって、フォーマットされた文書を作成できます。

すべてのサービス・タイプのうち、出力を制御しやすいのは RAW です。ただし、必要なタグをすべて明示的に出力しなければならないので必要な作業が増えます。XML サービスの出力は、サービスの文に FOR XML 句を適用することにより調節できます。SOAP サービスの出力は、CREATE 文または ALTER SERVICE 文の FORMAT 属性を適用することにより調節できます。

詳細については、『ASA SQL リファレンス・マニュアル』>「CREATE SERVICE 文」を参照してください。

文

文とはコマンドのことであり、通常はストアド・プロシージャです。文は、ユーザがサービスにアクセスしたときに呼び出されます。文を定義する場合、その文のみがそのサービスで実行可能になります。文は SOAP サービスには必須で、DISH サービスでは無視されます。デフォルトは NULL で、文がないことを示します。

文を含まないサービスを作成できます。文は URL から取得されます。このように設定されたサービスは、サービスをテストしたり、情報への一般的なアクセスが必要であったりする場合に便利です。文を含まないサービスを作成するには、文を全く省略するか、文の箇所に AS NULL というフレーズを使用します。

文のないサービスでは、Web クライアントが任意のコマンドを実行することを許可するため、深刻なセキュリティ上のリスクを負うことになります。そのようなサービスを作成する場合、認証を有効にして、強制的にすべてのクライアントが有効なユーザ名とパスワードを入力するようにしてください。こうすれば、文を定義したサービスのみが運用システムで実行されるようになります。

属性

使用可能な属性は、次のとおりです。一般的に、すべてオプションです。ただし、相互依存型のものもあります。

- **AUTHORIZATION** この属性は、サービスを使用できるユーザを制御します。デフォルト設定は ON です。文がない場合、認証は ON にしてください。また、認証設定は、USER 属性で定義されたユーザ名の解釈方法に影響します。

- **SECURE** ON に設定すると、安全な接続のみが許可されます。HTTP ポートが受信するすべての接続が自動的に HTTPS ポートにリダイレクトされます。デフォルトは OFF で、データベース・サーバを起動するときにこれらのポートが適切なオプションを使用して有効になっていれば、HTTP 要求と HTTPS 要求の両方が有効です。

- **USER** USER 句は、サービス要求を処理するのに使用できるデータベース・ユーザ・アカウントを制御します。ただし、この設定の解釈は、認証が ON か OFF かに依存します。

  認証が ON に設定されている場合、すべてのクライアントは接続時に有効なユーザ名とパスワードを入力する必要があります。認証が ON の場合、USER オプションは NULL、データベース・ユーザ名、データベース・グループ名のいずれかです。NULL の場合、どのデータベース・ユーザも接続して要求できます。要求は、そのユーザのアカウントとパーミッションを使用して実行されます。グループ名が指定されている場合、グループに

属するユーザのみが要求を実行できます。他のデータベース・ユーザはすべて、サービスを使用するためのパーミッションが拒否されます。

認証が OFF の場合は、文を指定する必要があります。また、ユーザ名も指定してください。すべての要求は、そのユーザのアカウントとパーミッションを使用して実行されます。したがって、サーバがパブリックなネットワークに接続されている場合、悪意の使用による損傷を制限するために、指定されたユーザ・アカウントのパーミッションを最小限にしてください。

- **GROUP**　DISH サービスのみに適用する GROUP 句は、DISH サービスで公開される SOAP サービスを決定します。DISH サービスによって公開される SOAP サービスは、その DISH サービスのグループ名で始まる名前を持つもののみです。したがってグループ名が、公開された SOAP サービスに共通するプレフィックスとなります。たとえば、GROUP *xyz* を指定した場合、SOAP サービス *xyz*/*aaaa*、*xyz*/*bbbb*、または *xyz*/*cccc* が公開されますが、*abc/aaaa* または *xyzaaaa* は公開されません。グループ名を指定しな場合、DISH サービスはデータベース内のすべての SOAP サービスを公開します。グループ名にはサービス名と同じ文字を使用できます。

  SOAP サービスは、複数の DISH サービスによって公開できます。具体的には、この機能によって、単一の SOAP サービスが複数のフォーマットでデータを提供することが可能になります。SOAP サービスで指定がない限り、サービス・タイプは DISH サービスから継承されます。したがって、フォーマット・タイプを宣言しない SOAP サービスを作成してから、それぞれ異なるフォーマットを指定する複数の DISH サービスに含めることができます。

- **FORMAT**　DISH サービスと SOAP サービスに適用する FORMAT 句は、SOAT または DISH の応答の出力フォーマットを制御します。.NET または Java JAX-RPC など、さまざまな種類の SOAP クライアントと互換性のある出力フォーマットが使用可能です。SOAP サービスのフォーマットが指定されていない場合は、フォーマットは DISH サービス宣言から継承されます。DISH サービスもフォーマットを宣言していない場合は、.NET クライアントと互換性がある DNET がデフォルトで使用さ

れます。フォーマットを宣言しない SOAP サービスは、複数の DISH サービスを定義することにより、それぞれ異なる FORMAT タイプを持つさまざまな種類の SOAP クライアントで使用できます。

- **URL** URL 句は URL の相互運用を制御し、XML、HTML、および RAW サービス・タイプのみの適用します。特に、URL パスを受け入れるか否か、また受け入れる場合はどのように処理するかを決定します。サービス名が文字 "/" で終了している場合は、URL を OFF に設定してください。

  詳細については、『ASA SQL リファレンス・マニュアル』>「CREATE SERVICE 文」を参照してください。

Web 要求を受信するデータベース・サーバの起動

データベース・サーバに HTTP または HTTPS で Web サービス要求を受信させたい場合は、サーバの起動時に受信する Web 要求のタイプをコマンド・ラインで指定する必要があります。デフォルトでは、データベース・サーバは Web 要求を受信しないため、クライアントはデータベースに定義されているサービスへアクセスする方法がありません。

また、どのポートで受信するかなど、HTTP または HTTPS サービスのさまざまなプロパティを、コマンド・ラインで指定することもできます。

また、データベース内に Web サービスを作成することも必要です。詳細については、『ASA データベース管理ガイド』> 「Web サービスの作成」を参照してください。

-xs オプションを使用してプロトコルを有効にできます。使用可能な Web サービス・プロトコルには、http と https の 2 つがあります。プロトコル名の後にオプションのパラメータをカッコに入れて追加すると、各タイプのサービスへのアクセスをカスタマイズできます。

オプションの一般的な構文は、次のとおりです。

-xs { *protocol* [**(***option*=*value*; …**)**], … }

プロトコル

次の Web サービス・プロトコル値を使用できます。

- **http** HTTP 接続を受信します。
- **https** HTTPS 接続を受信します。
- **https_fips** HTTPS FIPS 接続を受信します。
- **none** Web サービス要求を受信しません。これはデフォルト設定です。

オプション

使用可能なオプションは次のとおりです。

- **ServerPort [PORT]** Web 要求を受信するポート。デフォルトで、Adaptive Server Anywhere はポート 80 で HTTP 要求を受信し、ポート 443 でセキュア HTTP (HTTPS) 要求を受信します。

  たとえば、すでにポート 80 で動作中の Web サーバがある場合、次のオプションを使用してポート 8080 で Web 要求を受信するデータベース・サーバを起動できます。

  ```
  dbeng9 web.db -xs http(port=8080)
  ```

  別の例として、次のコマンドは SQL Anywhere のサンプル証明書を使用してセキュア・サーバを起動します。このコマンドは、1 行に入力してください。

  ```
  dbsrv9 -xs https(certificate=rsaserver.crt;
                   certificate_password=test)
  ```

警告
サンプル証明書は、テスト作業と開発作業にのみ使用します。この証明書は SQL Anywhere の標準部分であるため、保護機能は備えていません。アプリケーションを配備する前に独自の証明書で置換してください。

- **DatabaseName [DBN]** Web 要求を処理するときに使用するデータベースの名前を指定します。また、REQUIRED や AUTO キーワードを使用して URL 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。

  このパラメータが REQUIRED に設定されている場合は、URL 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、URL 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。URL 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。

  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。URL にはデータベース名を含めないでください。

- **LocalOnly [LOCAL]** このパラメータを YES に設定すると、ネットワーク・データベース・サーバは異なるマシンで実行中のクライアントからの通信をすべて拒否します。このオプションは、ほかのマシンからの Web サービス要求を受け入れることのないパーソナル・データベース・サーバには影響しません。デフォルト値は NO で、どの場所にあるクライアントの要求も受け入れます。

- **LogFile [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。

- **LogFormat [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。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 @T などのコードに置き換えられます。

  デフォルト値は **@T - @W - @I - @P - "@M @U @V" - @R - @L - @E** で、次のようなメッセージを生成します。

  ```
  06/15 01:30:08.114 - 0.686 - 127.0.0.1 - 80
    - "GET /web/show_table HTTP/1.1" - 200 OK - 55133
  -
  ```

  ログ・ファイルのフォーマットは Apache との互換性があるため、その分析に同じツールを使用できます。

- **LogOptions [LOPT]** ログ・ファイルに書き込まれるメッセージまたはメッセージのタイプを制御するキーワードとエラー番号を指定できます。

  詳細については、「LogOptions プロトコル・オプション [LOPT]」290 ページを参照してください。

使用可能なオプションの完全なリストと詳細については、「ネットワーク・プロトコル・オプション」276 ページを参照してください。

URL の解釈方法

URL (Universal Resource Locators) は、SOAP または HTTP Web サービスから取得できる HTML ページなどのドキュメントを指定します。Adaptive Server Anywhere で使用される URL は、Web のブラウズで見慣れた形式に従っています。データベース・サーバを介してユーザがブラウズする場合、それらの要求が従来のスタンドアロン Web サーバで処理されていないことを意識する必要はありません。

Adaptive Server Anywhere データベース・サーバは、フォーマットは標準ですが、URL の解釈方法は標準の Web サーバと異なります。データベース・サーバの起動時に指定するオプションも、その解釈方法に影響します。

URL の一般的な構文は、次のとおりです。

{ **http** | **https** } **://** [*user*:*password*@] *host* [**:***port*] [**/***dbn*] **/***service-name* [*parameters*] [**?***query*]

ユーザとパスワード

Web サービスに認証が必要な場合、ユーザ名とパスワードは、コロンで区切って電子メール・アドレスのようにホスト名の前に挿入することにより、URL の一部として直接渡すことができます。

ホストとポート

すべての標準的な HTTP 要求と同様に、URL はホスト名や IP 番号から始まり、オプションでポート番号になります。IP アドレスやホスト名とポートは、サーバが受信しているうちの 1 つにしてください。IP アドレスは、Adaptive Server Anywhere を実行しているマシンのネットワーク・カードのアドレスです。ポート番号は、データベース・サーバを起動したときに -xs オプションで指定したポート番号です。ポート番号を指定しない場合、そのサービス・タイプでデフォルトのポート番号が使用されます。たとえば、デフォルトでサーバはポート 80 で HTTP 要求を受信します。

詳細については、「ネットワーク・プロトコル・オプション」276 ページを参照してください。

データベース名

スラッシュの間にある次のトークンは、通常データベース名です。このデータベースはサーバで実行中であり、Web サービスが含まれている必要があります。

URL にデータベース名が表示されず、データベース名が -xs オプションの DBS 接続パラメータを使用して指定されていない場合は、デフォルトのデータベースが使用されます。

データベース・サーバで実行中のデータベースが 1 つだけか、またはデータベース名が -xs オプションの DBS 接続パラメータを使用して指定されている場合のみ、データベース名を省略できます。

サービス名

次の URL の部分はサービス名です。このサービスは、指定したデータベースに存在している必要があります。Web サービス名にスラッシュ文字が含まれていることもあるため、サービス名は次のスラッシュ文字にまたがる場合もあります。Adaptive Server Anywhere は、URL の残りの部分と定義されたサービスを一致させます。

URL にサービス名の指定がない場合は、データベース・サーバがサービスの指定されたルートを検索します。指定したサービスまたはルート・サービスが定義されていないと、サーバは `404 Not Found` エラーを返します。

パラメータ

対象となるサービスの種類によって、パラメータを指定する方法が異なります。HTML、XML、RAW サービスに対するパラメータは、次のいずれかの方法で渡すことができます。

- URL に追加
- 明示的なパラメータ・リストとして指定
- POST 要求の POST データとして指定

SOAP サービスに対するパラメータは、標準 SOAP 要求の一部として含める必要があります。これ以外の方法で提供される値は無視されます。

パラメータ値にアクセスするには、パラメータに名前を指定します。これらのホスト変数名にはプレフィクスとしてコロン (:) が付き、Web サービス定義の一部を形成する文に含めることができます。

たとえば、プロシージャ place_order を呼び出す文には、製品 ID 番号と数量パラメータが必要です。

```
call place_order( :product, :quantity )
```

パラメータには、HTTP_VARIABLE 関数を使用してもアクセスできます。詳細については、『ASA SQL リファレンス・マニュアル』>「HTTP_VARIABLE 関数 [HTTP]」を参照してください。

クエリ

クエリは、*name*=*value* の対を & 文字で区切ったリストです。このリストは、疑問符から始まります。GET 要求はこのような形でフォーマットされます。名前付き変数がある場合、対応する値が定義され、割り当てられます。

URL PATH を ON または ELEMENTS に設定すると、追加の変数が定義されます。ただし、この両者は独立しています。PATH を ON、または ELEMENTS、あるいはその両方に設定すると、要求された URL で変数を使用できます。

変数の詳細については、「変数の使用」348 ページを参照してください。

SOAP および DISH Web サービスの作成

SOAP および Web サービスは、Microsoft .NET や Java JAX-RPC で記述された SOAP クライアントなど、標準の SOAP クライアントがアクセスできる 標準の SOAP Web サービスを作成するための手段です。

SOAP サービス

SOAP サービスは、標準の SOAP 要求を受け入れ、処理する Web サービスを SQL Anywhere で構築するためのメカニズムです。

SOAP サービスを宣言するには、そのサービスが SOAP のタイプであることを指定します。標準の SOAP 要求の本文は、SOAP エンベロープで、特定のフォーマットの XML ドキュメントを意味します。Adaptive Server Anywhere は、指定したプロシージャを使用してこれらの要求を解析し、処理します。応答は、SOAP エンベロープでもある標準の SOAP 応答の形式で自動的にフォーマットされ、クライアントに返されます。

SOAP サービスの作成に使用する文の構文は、次のとおりです。

CREATE SERVICE *service-name*
    TYPE 'SOAP'
    [**FORMAT** { **'DNET'** | **'CONCRETE'** | **'XML'** | **NULL** }]
    [*common-attributes*]
    AS *statement*

DISH サービス

DISH サービスは、SOAP サービスのグループのプロキシとして機能するほか、現在公開している SOAP サービスを記述する WSDL ドキュメントを構築します。

DISH サービスを作成する場合、GROUP 句で指定された名前によって、その DISH サービスが公開する SOAP サービスが決定されます。DISH サービスの名前がプレフィックスとなっている名前を持つ SOAP サービスがすべて公開されます。たとえば、GROUP xyz を指定すると、SOAP サービス xyz/aaaa、xyz/bbbb、または xyz/cccc が公開されます。abc/aaaa や xyzaaaa という名前の SOAP サービスは公開されません。SOAP サービスは、複数の DISH サービスによって公開できます。グループ名を指定しな場合は、DISH サービスはデータベース内のすべての SOAP サービスを公開します。グループ名にはサービス名と同じ文字を使用できます。

DISH サービスの作成に使用する文の構文は、次のとおりです。

CREATE SERVICE *service-name*
    TYPE 'DISH'
    [**GROUP** { *group-name* | **NULL** }]
    [**FORMAT** { **'DNET'** | **'CONCRETE'** | **'XML'** | **NULL** }]
    [*common-attributes*]

SOAP および DISH サービスのフォーマット

CREATE SERVICE 文の FORMAT 句により、.NET や Java JAX-RPC など、さまざまな種類の SOAP クライアントに合わせて SOAP サービス・データ・ペイロードがカスタマイズされます。FORMAT 句は、DISH サービスによって返される WDSL ドキュメントの内容と、SOAP 応答によって返されるデータ・ペイロードのフォーマットに影響します。

デフォルト・フォーマットの DNET は、.NET DataSet フォーマットを必要とする .NET SOAP クライアント・アプリケーションで使用するネイティブ・フォーマットです。

CONCRETE フォーマットは、返されたデータ構造のフォーマットに基づいてインタフェースを自動的に生成する Java JAX-RPC や .NET などのクライアント用です。このフォーマットを指定すると、Adaptive Server Anywhere によって返された WSDL ドキュメントは結果セットを具体的に説明する ASADataset 要素を公開します。この要素は、ローの配列から構成されるローセットの包含階層で、それぞれにカラム要素の配列が含まれます。

XML フォーマットは、SOAP 応答を 1 つの大きな文字列として受け入れ、XML パーサによって必要な要素と値を検索して抽出する SOAP クライアントで使用します。通常このフォーマットは、さまざまな種類の SOAP クライアント間で最も手軽です。

SOAP サービスのフォーマットが指定されていない場合は、フォーマットは DISH サービス宣言から継承されます。DISH サービスもフォーマットを宣言していない場合は、.NET クライアントと互換性がある DNET がデフォルトで使用されます。フォーマットを宣言しない SOAP サービスは、複数の DISH サービスを定義することにより、それぞれ異なる FORMAT タイプを持つさまざまな種類の SOAP クライアントで使用できます。

同種の DISH サービスの作成

SOAP サービスでは、フォーマット・タイプを指定する必要はありません。つまり、フォーマット・タイプを NULL に設定します。このようにすると、SOAP サービスのプロキシとして機能する DISH サービスからフォーマットが継承されます。各 SOAP サービスに対して複数の DISH サービスがプロキシの役割をすることができ、それらの DISH サービスは同じ種類でなくてもかまいません。このことは、.NET や Java JAX-RPC など、さまざまな種類の SOAP クライアントで単一の SOAP サービスを使用できることを意味します。DISH サービスは、同じ SOAP サービスに対してフォーマットは異なっても同じデータ・ペイロードを公開するため、*同種*であると見なされます。

たとえば、以下の 2 つの SOAP サービスを考えてみます。どちらもフォーマットを指定していません。

```
CREATE SERVICE "abc/hello"
 TYPE 'SOAP'
 AS CALL hello(:student)

CREATE SERVICE "abc/goodbye"
 TYPE 'SOAP'
 AS CALL goodbye(:student)
```

これらのサービスはどちらも FORMAT 句を含んでいないため、フォーマットはデフォルトで NULL になります。したがって、プロキシとして機能している DISH サービスからフォーマットが継承されます。ここで、次のような DISH サービスを考えてみます。

```
CREATE SERVICE "abc_xml"
 TYPE 'DISH'
 GROUP "abc"
 FORMAT 'XML'

CREATE SERVICE "abc_concrete"
 TYPE 'DISH'
 GROUP "abc"
 FORMAT 'CONCRETE'
```

どちらの DISH サービスも同じグループ名 "abc" を指定しているため、両方が同じ SOAP サービス (主にプレフィックス "abc/" が付いている名前を持つ SOAP サービスすべて) のプロキシとして機能します。

ただし、abc_xml DISH サービスを介して 2 つの SOAP サービスのいずれかがアクセスされた場合、SOAP サービスは XML フォーマット・タイプを継承し、abc_concrete SOAP サービスを介してアクセスされると、CONCRETE フォーマットを継承します。

同種の DISH サービスは、作成した SOAP Web サービスに複数種類の SOAP クライアントがアクセスできるようにしたい場合、重複するサービスを避ける手段を提供しています。

チュートリアル：Java JA-RPC からの Web サービスへのアクセス

次のチュートリアルでは、Java JA-RPC から Web サービスにアクセスする方法を示します。JAX-RPC から SQL Anywhere SOAP Web サービスにアクセスするには、サービスが CONCRETE フォーマットであることを宣言します。

❖ SOAP および DISH サービスの作成

1　コマンド・プロンプトで、次のコマンドを実行し、このチュートリアルで使用する新しいデータベースを作成します。

```
dbinit jaxrpc.db
```

2　この新しいデータベースを起動します。データベース・サーバで HTTP 要求を受信し処理するように、-xs http フラグを使用します。-xs http(port=80) スイッチは、HTTP 要求を受け入れるようにデータベースに指示します。ポート 80 ですでに Web サーバが実行されている場合、このチュートリアルには 8080 などの別のポート番号を使用します。

```
dbsrv9 -xs http(port=80) jaxrpc.db
```

3　Interactive SQL を起動します。ユーザ名 DBA とパスワード SQL を使用して Web データベースに接続します。次の文を実行します。

a.　SOAP サービスで使用する簡単な hello プロシージャを作成します。

```
CREATE PROCEDURE "hello_proc"(IN user_name
char(64))
 RESULT (result_out long varchar)
 BEGIN
    SELECT 'hello' || ' ' || user_name || '¥n';
 END
```

b.　hello ストアド・プロシージャを呼び出す SOAP サービスを定義します。

```
CREATE SERVICE "sample/hello"
 TYPE 'SOAP'
 AUTORIZATION OFF
 SECURE OFF
 USER DBA
 AS CALL hello_proc(:user_name)
```

認証はオフになっているため、ユーザ名とパスワードを入力せずに誰でもこのサービスを利用できます。コマンドはユーザ DBA として実行されます。この方法は簡単ですが、安全性に優れていません。

c. SOAP サービスのプロキシとして機能する DISH サービスを作成し、WSDL を生成します。

```
CREATE SERVICE "sample/dish_service"
 TYPE 'DISH'
 GROUP "sample"
 FORMAT 'CONCRETE'
 AUTHORIZATION OFF
 SECURE OFF
 USER DBA
```

SOAP サービスと DISH サービスは、CONCRETE タイプです。この例では、SOAP サービスの作成時に FORMAT 句が省略されています。その結果、SOAP サービスは DISH サービスから CONCRETE フォーマットを継承します。

4 DISH サービスにより自動的に生成される WSDL を見てみます。そのためには、Web ブラウザを開き、URL (http://localhost:80/jaxrpc/sample/dish_service) を参照します。DISH サービスは次の WSDL ドキュメントを自動的に作成します。

このサービスのフォーマットは CONCRETE であるため、特に公開された ASADataset オブジェクトをよく見てください。この後の手順で、wscompile.bat アプリケーションはこの情報を使用して、これらのサービス用の SOAP 1.1 クライアント・インタフェースを生成します。

```
<?xml version="1.0" ?>
 <definitions xmlns:http="http://
schemas.xmlsoap.org/wsdl/http/" xmlns:soap="http://
schemas.xmlsoap.org/wsdl/soap/" xmlns:s="http://
www.w3.org/2001/XMLSchema" xmlns:soapenc="http://
schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/
textMatching/" xmlns:mime="http://
schemas.xmlsoap.org/wsdl/mime/" name="sample/
dish_service" targetNamespace="http://localhost/
jax/sample/dish_service" xmlns:s3="http://
localhost/jax/sample/dish_service" xmlns="http://
schemas.xmlsoap.org/wsdl/">
  <types>
   <s:schema attributeFormDefault="qualified"
elementFormDefault="qualified"
targetNamespace="http://localhost/jax/sample/dish_service">
    <s:import namespace="http://www.w3.org/2001/
XMLSchema" />
    <s:complexType name="ASADataset">
     <s:sequence>
      <s:element name="rowset">
       <s:complexType>
        <s:sequence>

<s:element name="row" minOccurs="0"
maxOccurs="unbounded">
          <s:complexType>
           <s:sequence>
            <s:any minOccurs="0"
maxOccurs="unbounded"/>
           </s:sequence>
          </s:complexType>
         </s:element>
        </s:sequence>
       </s:complexType>

</s:element>
     </s:sequence>
    </s:complexType>
    <s:element name="error" type="s:string" />
    <s:element name="hello">
```

```
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="0" maxOccurs="1"
name="user_name" nillable="true" type="s:string" />
      </s:sequence>

</s:complexType>
    </s:element>
    <s:element name="helloResponse">
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="1" maxOccurs="1"
name="helloResult" type="s3:ASADataset" />
       <s:element name="sqlcode" type="s:int" />
      </s:sequence>
     </s:complexType>

</s:element>
   </s:schema>
  </types>
  <message name="helloIn">
   <part name="parameters" element="s3:hello" />
  </message>
  <message name="helloOut">
   <part name="parameters"
element="s3:helloResponse" />
  </message>

<message name="ASAFaultMessage">
     <part name="error" element="s3:error" />
    </message>
  <portType name="sample/dish_serviceSoapPort" >
   <operation name="hello">
    <input message="s3:helloIn" />
    <output message="s3:helloOut" />
   <fault message="s3:ASAFaultMessage" name="error"
/>
   </operation>

</portType>
  <binding name="sample/dish_serviceSoapBinding"
type="s3:sample/dish_serviceSoapPort">
   <soap:binding style="document" transport="http://
schemas.xmlsoap.org/soap/http" />
   <operation name="hello">
    <soap:operation soapAction="http://localhost/
```

```
jax/sample/hello" style="document" />
    <input>
     <soap:body use="literal" />
    </input>
    <output>

<soap:body use="literal" />
    </output>
    <fault name="error">
     <soap:fault name="error" use="literal" />
    </fault>
   </operation>
  </binding>
 <service name="sample/dish_service">
  <port name="sample/dish_serviceSoap"
binding="s3:sample/dish_serviceSoapBinding">

<soap:address location="http://localhost/jax/
sample/dish_service" />
   </port>
  </service>
 </definitions>
```

このチュートリアルの次の項では、Java からこれらのサービスにアクセスします。そのためには、Sun の wscompile ツールを使用する必要があります。

wscompile を含む Java JAX-RPC ツールをダウンロードするには、http://java.sun.com/xml/downloads/jaxrpc.html を参照してください。

❖ JAX-RPC インタフェースを生成し Web サービスで使用するには、次の手順に従います。

1　最初の手順は、wscompile に指示する簡単な XML ドキュメントを作成することです。この構成ドキュメントのサンプルは、Sun から入手できます。wsdl 要素の場所属性を DISH サービスの URL で置換する必要があるだけです。オプションで、パッケージ・ファイル名も指定できます。パッケージ名により、生成された .java ファイルと .class ファイルは同じ名前のサブディレクトリに格納されます。

テキスト・エディタを使用して、次の内容の *config.xml* という XML ドキュメントを作成します。

```
<?xml version="1.0" encoding="UTF-8"?>
 <configuration
     xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/
config">
     <wsdl
         location="http://localhost:80/jaxrpc/
sample/dish_service"
         packageName="asa" />
 </configuration>
```

この場合、指定の場所は DISH サービスの URL です。さらに、生成されたすべてのファイルが *asa* という新しいサブディレクトリに格納されるように、オプションの packageName 属性が追加されました。

2 コマンド・プロンプトで次のコマンドを入力します。

```
wscompile -gen -keep config.xml
```

-gen フラグは、指定の URL から WSDL ドキュメントを取り出し、そのためのインタフェースをコンパイルするよう wscompile に指示します。-keep フラグは、java ファイルを削除しないよう wscompile に指示します。このフラグがない場合は、対応する .class ファイルの生成後にこれらのファイルは削除されます。これらのファイルを保存すると、インタフェースの構成の検査が簡単になります。

wscompile コマンドが見つからない場合は、マシンに wscompile があること、およびそれがパス上にあることを確認します。

このコマンドが完了すると、それぞれコンパイルされた .class バージョンとともに、以下の Java ファイルを含む *asa* というサブディレクトリが検出されます。

- ASADataset.java
- ASADataset_LiteralSerializer.java
- ASAFaultMessage.java
- Dish_service.java
- Dish_serviceSoapPort.java
- Dish_serviceSoapPort_Stub.java
- Dish_serviceSoapPort_hello_Fault_SOAPBuilder.java

- Dish_serviceSoapPort_hello_Fault_SOAPSerializer.java
- Dish_service_Impl.java
- Dish_service_SerializerRegistry.java
- Hello.java
- HelloResponse.java
- HelloResponse_LiteralSerializer.java
- Hello_LiteralSerializer.java
- Row.java
- Row_LiteralSerializer.java
- Rowset.java
- Rowset_LiteralSerializer.java

3　次の Java ソース・コードをテキスト・ファイルに保存してコンパイルします。

```
// HelloTest.java illustrates a web service client
 // that calls the dish_service and prints out the
data:

import java.util.*;
 import asa.*;

public class HelloTest
 {
     public static void main( String[] args )
     {
         try {
             Dish_service_Impl service = new
Dish_service_Impl();
             Dish_serviceSoapPort port =
service.getDish_serviceSoap();
         // this is the soap service call to hello...
            HelloResponse response = port.hello("New
User");
             ASADataset result =
response.getHelloResult();
             Rowset rowset = result.getRowset();
             Row[] row = rowset.getRow();

for ( int i = 0; i < row.length; i++ ) {
             // column data is contained as a
SOAPElement...
                 javax.xml.soap.SOAPElement[] col =
row[i].get_any();
```

```
                for ( int j = 0; j < col.length; j++
) {
                     System.out.print("<" +
col[j].getLocalName() + ">" +
                         col[j].getValue() );
                }
                System.out.println();
            }

}
         catch (Exception x) {
             x.printStackTrace();
         }
     }
 }
```

4 上記のアプリケーションを実行します。次の出力が生成されます。

```
<result_out>hello New User
```

Web サービス・クライアント関数とプロシージャの作成

Adaptive Server anywhere データベースは、Web サービスを提供すると同時に、Web サービスを消費します。Web サービスがクライアント・プロシージャまたは関数と同じデータベースに存在しない限り、それらはインターネットで利用できる標準の Web サービスの場合や、Adaptive Server Anywhere データベースによって提供される Web サービスの場合もあります。

Adaptive Server Anywhere は、HTTP と SOAP の両方の Web サービス・クライアントとして機能できます。この機能は、ストアド関数またはストアド・プロシージャにより提供されます。

クライアント関数とプロシージャは、以下の SQL 文を使用して作成し、操作します。

- 『ASA SQL リファレンス・マニュアル』>「CREATE FUNCTION 文」
- 『ASA SQL リファレンス・マニュアル』>「CREATE PROCEDURE 文」
- 『ASA SQL リファレンス・マニュアル』>「ALTER FUNCTION 文」
- 『ASA SQL リファレンス・マニュアル』>「ALTER PROCEDURE 文」
- 『ASA SQL リファレンス・マニュアル』>「DROP 文」

たとえば、Web サービス・クライアント関数の作成に使用される CREATE FUNCTION 文と CREATE PROCEDURE 文の構文は、次のとおりです。

CREATE PROCEDURE [*owner*.]*procedure-name* **(** [*parameter, …*] **)**
    URL *url-string*
    [*proc-attributes*]

CREATE FUNCTION [*owner*.]*procedure-name* **(** [*parameter, …*] **)**
    RETURNS *data-type*
    URL *url-string*
    [*proc-attributes*]

この構文の鍵となるのは、URL 句です。この句は、プロシージャでアクセスする Web サービスの URL を提供するために使用されます。URL 句の基本的な構文は、次のとおりです。

url-string :
'{**HTTP**|**HTTPS**}:**//**[*user*:*password*@]*hostname*[:*port*][**/***path*]'

オプションのユーザおよびパスワード情報により、認証を必要とする Web サービスにアクセスできます。ホスト名には、Web サービスを提供しているマシンの名前または IP アドレスを使用できます。

ポート番号は、サーバがデフォルト以外のポート番号で受信する場合のみ必要です。デフォルトのポート番号は、HTTP サーバの場合は 80 で、HTTPS サービスの場合は 443 です。

パスは、サーバ上のリソースまたは Web サービスを識別します。

要求は、別の Adaptive Server Anywhere データベースによって提供されたものか、インターネットで利用可能なものかにかかわらず、どの Web サービスにも送信できます。Web サービスは、同じデータベース・サーバによって提供されていてもかまいませんが、クライアント関数と同じデータベースにあるものは使用できません。同じデータベース内の Web サービスにアクセスしようとすると、「403 Forbidden」のエラーが返されます。

感嘆符は代入パラメータに使用されるため、プロシージャ定義の文字列のどこかに含まれる感嘆符はエスケープする必要があります。詳細については、「! 文字のエスケープ」343 ページを参照してください。

一般的な句

プロシージャ・コールに関する追加の詳細情報を提供するためのその他の句には、以下があります。

proc-attributes :
[**TYPE** { **'HTTP**[:{**GET**|**POST**}]' | **'SOAP**[:{**RPC**|**DOC**}]' }]
[**NAMESPACE** *namespace-string*]
[**CERTIFICATE** *certificate-string*]
[**CLIENTPORT** *clientport-string*]
[**PROXY** *proxy-string*]

TYPE 句は、Adaptive Server Anywhere に Web サービス・プロバイダへの要求のフォーマット方法を指定するために重要です。標準の SOAP タイプの RPC と DOC を使用できます。GET や POST などの標準の

HTTP メソッドも使用可能で、それぞれ HTTP:GET および HTTP:POST と指定されます。HTTP を指定すると、HTTP:POST を暗黙で指定したことになります。

タイプ SOAP が選択されると、Adaptive Server Anywhere は自動的に要求を SOAP 要求に必要な標準フォーマットの XML ドキュメントとしてフォーマットします。SOAP 要求は常に XML ドキュメントであるため、常にこの HTTP POST 要求が暗黙的に使用され、タイプ SOAP が選択されるたびに SOAP 要求ドキュメントをサーバに送信します。タイプ SOAP は、SOAP:RPC を暗黙で指定します。

Web サービス・クライアント関数とプロシージャの名前

出力 SOAP 要求の構築時には、プロシージャ名が SOAP 操作名として使用されます。さらに、パラメータの名前も SOAP 要求エンベロープのタグ名に表示されます。したがって、これらの名前を SOAP サーバで必要なとおりに正しく指定することは、SOAP ストアド・プロシージャの定義において重要なポイントです。これは、SOAP プロシージャと関数の名前には、SQL Anywhere のプロシージャ名と関数名に適用されるもの以外にも、制約が加えられることを意味します。

次のプロシージャ定義は、これを具体的に示しています。

```
CREATE PROCEDURE MyOperation (a INTEGER, b CHAR(128) )
 URL 'HTTP://localhost'
 TYPE 'SOAP:DOC'
```

次の文などにより、このプロシージャが呼び出されると、SOAP 要求が生成されます。

```
CALL MyOperation( 123, 'abc' )
```

プロシージャ名はこの例では MyOperation ですが、要求本文内の <m:MyOperation> タグ名に表示されます。さらに、プロシージャに対する 2 つのパラメータ a と b は、それぞれ <m:a> と <m:b> になります。

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
   xmlns:m="http://localhost">
   <SOAP-ENV:Body>
```

```
        <m:MyOperation>
          <m:a>123</m:a>
          <m:b>abc</m:b>
        </m:MyOperation>
      </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

ネームスペース URI

すべての SOAP 要求は、メソッド・ネームスペース URI を必要とします。サーバ側の SOAP プロセッサはこの URI を使用して、要求のメッセージ本文内にあるさまざまなエンティティの名前を解釈します。

SOAP:DOC または SOAP:RPC の SOAP 関数またはプロシージャを作成する場合、ネームスペース URI を指定しなければ呼び出しが成功しない可能性があります。必要なネームスペース値は、WSDL 記述ドキュメントやサービスのマニュアルから取得できます。NAMESPACE 句は、SOAP 関数とプロシージャのみに適用します。デフォルトのネームスペース値は、プロシージャの URI で、オプションのパス・コンポーネントとユーザおよびパスワードの値は含みません。

HTTPS 要求

セキュア HTTP 要求を発行するためには、クライアントはサーバの証明書、またはサーバの証明書の署名に使用する証明書にアクセスします。この証明書によって、Adaptive Server Anywhere で要求を暗号化する方法が指定されます。証明書の値は、安全化されていないサーバ宛ての要求をセキュア・サーバへリダイレクトする場合にも必要です。

証明書情報を提供するには、2 つの方法があります。証明書をファイルに保存してファイル名を指定する方法と、証明書全体を文字列値として提供する方法です。両方を実行することはできません。

証明書属性は、次のようにセミコロンで区切られた key=value ペアとして構成された文字列値として提供されます。

certificate-string :
{ **file=***filename* | **certificate=***string* } **; company=***company* **;**
unit=*company-unit* **; name=** *common-name*

次のキーを使用できます。

キー	省略形	説明
file		証明書のファイル名。

キー	省略形	説明
certificate	cert	証明書そのもの。Base64 形式でエンコードされています。
company	co	証明書で指定された会社。
unit		証明書で指定された会社単位。
name		証明書で指定された通称。

たとえば、次の文はクライアントと同じマシン上にある Web サービスへの安全な要求を作成します。

```
CREATE PROCEDURE test ()
 URL 'HTTPS://localhost/myservice'
 CERTIFICATE 'file=C:¥srv_cert.crt;co=iAnywhere;
              unit=ASA;name=JohnSmith'
```

TYPE 句が含まれていないため、要求は SOAP:RPC タイプであると見なされます。サーバのパブリック証明書は、*C:¥srv_cert.crt* ファイルにあります。

クライアント・ポート

ファイアウォールを介して Web サービスにアクセスする場合、サーバへの接続を確立するときに使用するポートを Adaptive Server Anywhere に対して指定する必要があることがよくあります。通常、ポート番号は動的に取得されるため、ファイアウォールによって特定範囲のポートへのアクセスが制限されていない限り、デフォルトの動作を使用してください。

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。次の例で示すように、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することができます。

```
CREATE PROCEDURE test ()
 URL 'HTTPS://localhost/myservice'
 CLIENTPORT '5040,5050-5060,5070'
```

ポート番号のリストまたは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、短いタイムアウト時間が発生します。その間、同じリモート・

サーバと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。

この機能は、ClientPort Network Protocol オプションと類似しています。詳細については、『ASA データベース管理ガイド』> 「ClientPort プロトコル・オプション [CPORT]」を参照してください。

プロキシの使用

プロキシ・サーバを使用して実行しなければならない Web サービス要求もあります。そのような場合は、PROXY 句を使用してプロキシ・サーバの URL を指定します。

値のフォーマットは、URL 句と同じですが、ユーザ、パスワード、パス値などは無視されます。

url-string :
'{HTTP|HTTPS}://[*user*:*password*@]*hostname*[:*port*][/*path*]'

プロキシを指定すると、Adaptive Server Anywhere は要求をフォーマットし、指定されたプロキシ URL を使用してそれをプロキシ・サーバに送ります。プロキシ・サーバは、最終送信先へ要求を転送し、応答を取得し、Adaptive Server Anywhere へそれを返します。

結果値と結果セット

Web サービス・クライアント呼び出しは、ストアド関数またはストアド・プロシージャのどちらでも実行できます。関数を使用した場合、戻り値のタイプは CHAR、VARCHAR、LONG VARCHAR などの文字データ型です。戻り値は、HTTP 応答の本文です。ヘッダー情報は含まれません。HTTP ステータス情報を含む要求に関する追加情報は、プロシージャによって返されます。したがって、この追加情報にアクセスする場合は、プロシージャの使用をおすすめします。

SOAP プロシージャ

SOAP 関数からは SOAP 応答を含んだ XML ドキュメントが返されます。

SOAP 応答は構造化された XML ドキュメントであるため、デフォルトでは Adaptive Server Anywhere はこの情報を利用してさらに役立つ結果セットを作成しようとします。返された応答ドキュメント内の最

上位レベルの各タグが抽出され、カラム名として使用されます。これらのタグのそれぞれの下にあるサブツリーの内容は、そのカラムのローの値として使用されます。

たとえば、次の SOAP 応答が返される場合、Adaptive Server Anywhere は以下のデータ・セットを作成します。

```
<SOAP-ENV:Envelope
   xmlns:SOAPSDK1="http://www.w3.org/2001/XMLSchema"
   xmlns:SOAPSDK2="http://www.w3.org/2001/XMLSchema-
instance"
   xmlns:SOAPSDK3="http://schemas.xmlsoap.org/soap/
encoding/"
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/">
   <SOAP-ENV:Body>
      <ElizaResponse xmlns:SOAPSDK4="SoapInterop">
         <Eliza>Hi, I'm Eliza. Nice to meet you.</Eliza>
      <ElizaResponse>
   </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
```

Eliza
Hi, I'm Eliza. Nice to meet you.

この例では、応答ドキュメントは SOAP-ENV:Body タグ内にある ElizaResponse タグによって区切られています。

結果セットには、最上位レベルのタグの数だけのカラムが含まれます。SOAP 応答には最上位レベルのタグが 1 つしかないため、この結果セットのカラムは 1 つだけです。この最上位レベル・タグである Eliza が、カラム名となります。

XML 処理機能

SOAP 応答を含む XML 結果セット内の情報には、組み込みの Open XML 処理機能を使用してもアクセスできます。

次の例では、OPENXML 機能を使用して SOAP 応答の一部を抽出します。この例は、SYSWEBSERVICE テーブルの内容を公開するために SOAP サービスとして Web サービスを使用しています。

```
CREATE SERVICE get_webservices
 TYPE 'SOAP'
 AUTHORIZATION OFF
 USER DBA
 AS SELECT * FROM SYSWEBSERVICE
```

2 番目の Adaptive Server Anywhere データベースで作成する次のストアド関数は、この webservice への呼び出しを発行します。この関数の戻り値は、SOAP 応答ドキュメント全体です。DNET がデフォルトの SOAP サービス・フォーマットであるため、応答は .NET DataSet フォーマットになります。

```
CREATE FUNCTION get_webservices ()
 RETURNS LONG VARCHAR
 URL 'HTTP://localhost/get_webservices'
 TYPE 'SOAP:DOC'
```

次の文は、結果セットの 2 つのカラムを OPENXML 関数を使用して抽出する方法を示しています。それらは *service_name* カラムと *secure_required* カラムで、SOAP サービスがセキュアで、HTTPS を必要とすることを示しています。

```
SELECT *
 FROM OPENXML( get_webservice(), '//row' )
 WITH ("Name"    char(128) 'service_name',
       "Secure?" char(1)   'secure_required' )
```

この文は、ロー・ノードの子孫を選択することによって機能します。WITH 句は、目的の 2 つの要素に基づき、結果セットを作成します。*get_webservices* サービスのみが存在すると想定し、この関数は以下の結果セットを返します。

Name	**Secure?**
get_webservices	N

Adaptive Server Anywhere で使用できる XML 処理機能の詳細については、『ASA SQL ユーザーズ・ガイド』> 「データベースにおける XML の使用」を参照してください。

その他のタイプのプロシージャ

その他のタイプのプロシージャは、応答に関する全情報を 2 つのカラムから成る結果セットで返します。この結果セットには、応答ステータス、ヘッダ情報、および本文が含まれます。最初のカラムには *Attribute*、2 番目のカラムには *Value* という名前が付けられています。どちらも LONG VARCHAR データ型です。

結果セットには、応答ヘッダ・フィールドごとに 1 ロー、HTTP ステータス行 (*Status* 属性) に対して 1 ロー、応答本文 (*Body* 属性) に対して 1 ローが含まれます。

次の例は、一般的な応答を示します。

Attribute	**Value**
Status	HTTP /1.0 200 OK
Body	<!DOCTYPE HTML … ><HTML> … </HTML>
Content-Type	text/html
Server	GWS/2.1
Content-Length	2234
Date	Mon, 18 Oct 2004, 16:00:00 GMT

結果セットからの選択

SELECT 文を使用して、結果セットから値を取り出します。取り出した値はテーブルに保存したり、変数を設定したりするために使用します。

```
CREATE PROCEDURE test( INOUT parm CHAR(128) )
 URL 'HTTP://localhost/test'
 TYPE 'HTTP'
```

このプロシージャは、HTTP タイプであるため、前の項で説明した 2 つのカラムから成る結果セットを返します。最初のカラムは属性名、2 番目のカラムは属性値です。キーワードは、HTTP 応答ヘッダ・フィールドにあるものと同様です。Body 属性には、メッセージの本文が含まれ、これは通常 HTML ドキュメントです。

以下のように結果セットをテーブルに挿入する方法があります。

```
CREATE TABLE StoredResults(
       Attribute LONG VARCHAR,
       Value        LONG VARCHAR
 )
```

結果セットは、次のようにこのテーブルに挿入します。

```
INSERT INTO StoredResults SELECT *
 FROM test('Storing into a table')
 WITH (Attribute LONG VARCHAR, Value LONG VARCHAR)
```

SELECT 文の通常の構文に従い、句を追加できます。たとえば、結果セットの特定のローのみが必要な場合は、WHERE 句を追加して SELECT の結果を 1 つのローに限定することができます。

```
SELECT Value
 FROM test('Calling test for the status code')
 WITH (Attribute LONG VARCHAR, Value LONG VARCHAR)
 WHERE Attribute = 'Status'
```

この文は、結果セットからステータス情報のみを選択します。このようにして、この文は呼び出しが成功したことを確認するために使用できます。

パラメータ

Web サービス・クライアントとして機能するストアド関数とストアド・プロシージャは、その他の関数やプロシージャと同様にパラメータを使用して宣言できます。パラメータの代入中に使用する場合を除き、これらのパラメータ値は HTTP 要求または SOAP 要求の一部として渡されます。

加えて、パラメータはストアド関数またはストアド・プロシージャの呼び出し時に、それらの本文内のプレースホルダを置換するためにも使用できます。特定の変数のプレースホルダが存在しない場合、パラメータとその値が要求の一部として渡されます。このようにして代入に使用されるパラメータは、Web サービス要求の一部として渡されません。

渡されるパラメータ

パラメータの代入中に使用する場合を除き、関数またはプロシージャのすべてのパラメータは、Web サービス要求の一部として渡されます。渡されるときのフォーマットは、Web サービス要求のタイプによって異なります。

HTTP 要求

HTTP:GET タイプの要求のパラメータは、URL でエンコードされます。たとえば、次のプロシージャは 2 つのパラメータを宣言します。

```
CREATE PROCEDURE test ( a INTEGER, b CHAR(128) )
 URL 'HTTP://localhost/myservice'
 TYPE 'HTTP:GET'
```

123 と "xyz" という値を使ってこのプロシージャを呼び出す場合、要求に使用する URL は次に示したものと同等になります。

```
HTTP://localhost/myservice?a=123&b=xyz
```

タイプが HTTP:POST である場合、パラメータとその値が要求の本文の一部になります。2 つのパラメータと値の場合、次のテキストがヘッダの後、HTTP 要求の本文に表示されます。

```
a=123&b=xyz
```

SOAP 要求

SOAP 要求に渡されたパラメータは、SOAP 仕様で指定されているように、要求本文の一部としてひとまとめにされます。

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
   xmlns:m="http://localhost">
   <SOAP-ENV:Body>
     <m:test>
       <m:a>123</m:a>
       <m:b>abc</m:b>
     </m:test>
   </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
```

代入パラメータ

ストアド・プロシージャまたはストアド関数の宣言済みパラメータは、そのプロシージャまたは関数が実行されるたびに、ストアド関数またはストアド・プロシージャ定義内のプレースホルダを自動的に置き換えます。感嘆符 (!) の後に宣言されたパラメータの 1 つの名前が続いている部分文字列はすべて、パラメータの値で置換されます。

たとえば、次のプロシージャ定義では、URL 全体がパラメータとして渡されます。このプロシージャが呼び出されるたびに、異なる値を使用できます。

```
CREATE PROCEDURE test ( url CHAR(128) )
 URL '!url'
 TYPE 'HTTP:POST'
```

たとえば、次のようなプロシージャを使用できます。

```
CALL test ( 'HTTP://localhost/myservice' )
```

ユーザとパスワード値の非表示

代入パラメータの有効な適用例として、ユーザ名やパスワードなどの機密の値を Web サービス・クライアント関数やプロシージャの定義の一部にすることを避けることがあります。そのような値がプロシージャまたは関数定義でリテラルとして指定されていると、それらはシステム・テーブルに保存され、データベースのすべてのユーザによって簡単にアクセスされてしまいます。このような値をパラメータとして渡すと、この問題を回避できます。

たとえば、次のプロシージャ定義には、ユーザ名とパスワードがプロシージャ定義の一部としてプレーン・テキストで含まれています。

```
CREATE PROCEDURE test
 URL 'HTTP://DBA:SQL@localhost/myservice'
```

この文字列は、システム・テーブルにプレーン・テキストとして表示されるため、認証値に簡単にアクセスできてしまいます。ユーザとパスワードをパラメータとして宣言するとこの問題を避けることができます。これにより、ユーザとパスワードの値はプロシージャが呼び出されたときにしか提供されなくなります。

```
CREATE PROCEDURE test ( uid CHAR(128), pwd CHAR(128) )
 URL 'HTTP://!uid:!pwd@localhost/myservice'
```

このプロシージャは次のように呼び出されます。

```
CALL test ( 'DBA', 'SQL' )
```

別の例として、代入パラメータを使用してファイルからストアド・プロシージャまたはストアド関数に暗号化証明書を渡すことができます。

```
CREATE PROCEDURE secure( cert LONG VARCHAR )
 URL 'https://localhost/secure'
 TYPE 'HTTP:GET'
 CERTIFICATE
'cert=!cert;company=test;unit=test;name=RSA Server'
```

このプロシージャを呼び出すときに証明書を文字列として提供します。次の呼び出し例では、証明書をファイルから読み出します。証明書は、CERTIFICATE 句の **file=** キーワードを使用して、ファイルから直接読み出すことができるため、これは説明のためにのみ行います。

```
CALL secure(
xp_read_file('C:¥asainstalldir¥win32¥rsaserver.crt') )
```

! 文字のエスケープ

感嘆符 (!) は Web サービス・クライアントのストアド関数とストアド・プロシージャのコンテキストで、代入パラメータのプレースホルダを識別するために使用するため、プロシージャの属性文字列の一部としてこの文字を含める場合は、エスケープします。そのためには、感嘆符にプレフィックスとしてもう 1 つの感嘆符を付けます。これにより、Web サービス・クライアントまたは Web サービス関数定義内のすべての "!!" が、"!" で置換されます。

プレースホルダとして使用されたパラメータ名には、アルファベット文字のみを含めます。さらに、あいまいにならないように、プレースホルダの後にはアルファベット以外の文字を挿入します。一致するパラメータ名のないプレースホルダは、自動的に削除されます。たとえば、次のプロシージャでは、パラメータ size はプレースホルダを置換しません。

```
CREATE PROCEDURE orderitem ( size CHAR(18) )
 URL 'HTTP://salesserver/order?size=!sizeXL'
 TYPE 'SOAP:RPC'
```

!sizeXL は、一致するパラメータのない有効なプレースホルダであるため、常に削除されます。

パラメータのデータ型変換

文字またはバイナリ・データ型でないパラメータ値は、要求に追加する前に文字列表現に変換されます。この処理は、値を文字型にキャストすることに相当します。変換は、関数またはプロシージャの呼び出し時に、データ型のフォーマット・オプションの設定に従って行われます。具体的には、変換は PRECISION、SCALE、TIMESTAMP_FORMAT などのオプションによって影響されます。

HTML ドキュメントを提供するプロシージャ

一般的に、特定のサービスに送信される要求を処理するプロシージャを記述するのが最も簡単です。このようなプロシージャは Web ページを返します。オプションで、プロシージャは出力をカスタマイズするために、URL の一部として渡される引数を受け入れることができます。

しかし、次の例はさらに単純です。これはサービスの単純さを表しています。この Web サービスは "Hello world!" というフレーズを返すだけです。

```
CREATE SERVICE hello TYPE 'RAW'
       AUTHORIZATION OFF USER DBA
       AS select 'Hello world!'
```

Web 要求の処理を可能にするために -xs オプションと共にサーバを起動し、任意の Web ブラウザから URL **http://localhost/hello** を要求します。Hello world! という言葉が無地のページに表示されます。

HTML ページ

上記のページは、使用しているブラウザにプレーン・テキストで表示されます。これは、デフォルトの HTTP コンテンツタイプが text/plain であるためです。HTML でフォーマットされたごく普通の Web ページを作成するには、次の 2 つの作業を行ってください。

- HTTP コンテンツタイプ・ヘッダ・フィールドを text/html に設定して、ブラウザが HTML を予期するようにします。
- 出力に HTML タグを含めます。

出力にタグを書き込むには 2 つの方法があります。1 つは、CREATE SERVICE 文で TYPE HTML フレーズを使用する方法です。この方法では、Adaptive Server Anywhere によって HTML タグが追加されます。これは、たとえばテーブルを返す場合などにうまく機能します。

もう 1 つは、TYPE RAW を使用して必要なタグをすべて自分で書き出す方法です。この 2 番目の方法は出力を最も制御できます。RAW タイプを指定しても、出力が必ずしも HTML または XML フォーマットではないという意味ではありません。これは、タグ自身を追加せずにクライアントに直接戻り値を渡せるということを Adaptive Server Anywhere に通知するだけです。

次のプロシージャでは、より凝ったバージョンの Hello world を生成します。便宜上、本文については次のプロシージャで扱いますが、これは Web ページをフォーマットするものです。

組み込みプロシージャ sa_set_http_header を使用して HTTP ヘッダ・タイプを指定するので、ブラウザは適切に結果を解釈します。この文を省略すると、ブラウザは、HTML コードをドキュメントのフォーマットに使用せず、すべての HTML コードを表示します。

```
CREATE PROCEDURE hello_pretty_world ()
 RESULT (html_doc long varchar)
 BEGIN
   call dbo.sa_set_http_header( 'Content-Type', 'text/
html' );
   select '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01/
/EN">¥n'
     || '<html>¥n'
     || '<head>¥n'
     || '  <title>Hello Pretty World</title>¥n'
     || '</head>¥n'
     || '<body>¥n'
     || '  <h1>Hello Pretty World!</h1>¥n'
     || '  <p>(If you see the tags in your browser,
check that¥n'
     || '      the Content-Type header is set to text/
html.)¥n'
     || '</body>¥n'
     || '</html>';
 END
```

次の文は、このプロシージャを使用するサービスを作成します。この文は、Hello Pretty World の Web ページを生成する上記のプロシージャを呼び出します。

```
CREATE SERVICE hello_pretty_world TYPE 'RAW'
        AUTHORIZATION OFF USER DBA
        AS call hello_pretty_world()
```

いったんプロシージャとサービスを作成したら、Web ページへのアクセスが可能になります。データベース・サーバが正しい -xs オプション値と共に起動していることを確認してください。これにより、URL **http://localhost/hello_pretty_world** を Web ブラウザで開けるようになります。

Hello Pretty World というタイトルの、単純な HTML ページでフォーマットされた結果が表示されます。Web ページをより凝ったものにするには、コンテンツを増やす、より多くのタグやスタイル・シートを使用する、ブラウザで実行するスクリプトを使用する、などを行ってください。どのような場合でも、ブラウザの要求を処理するためには必要なサービスを作成する必要があります。

組み込みのストアド・プロシージャの詳細については、『ASA SQL リファレンス・マニュアル』>「システム・プロシージャとカタログ・ストアド・プロシージャ」を参照してください。

ルート・サービス

URL にサービス名が含まれていない場合、Adaptive Server Anywhere は Web サービスの指定されたルートを検索します。ルート・ページの役割は、従来の多くの Web サーバにおける index.html ページの役割に似ています。

ルート・サービスは、Web サイトのアドレスのみを含む URL 要求を処理できるので、ホーム・ページの作成に便利です。たとえば、次のプロシージャとサービスは、URL **http://localhost** をブラウズした場合に表示される簡単な Web ページを実装しています。

```
CREATE PROCEDURE home_page
 RESULT (html_doc long varchar)
 BEGIN
   call dbo.sa_set_http_header( 'Content-Type', 'text/
html' );
   select '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01/
/EN">¥n'
       || '<html>¥n'
       || '<head>¥n'
       || '  <title>My Home Page</title>¥n'
       || '</head>¥n'
       || '<body>¥n'
       || '  <h1>My home on the web<h1>¥n'
       || '  <p>Thank you for visiting my web site!¥n'
       || '</body>¥n'
       || '</html>';
 END
```

次に、このプロシージャを使用するサービスを作成します。

```
CREATE SERVICE root TYPE 'RAW'
 AUTHORIZATION OFF USER DBA
 AS call home_page()
```

データベース・サーバを起動するときにデータベース名が必須であると指定しない限り、URL **http://localhost** をブラウズすることでこの Web ページにアクセスできます。

例

たとえば、クイック・スタートの例での要求には、データベース名もパラメータ名も含まれていません。この要求では、**/tables** がサービスを識別します。

```
http://localhost:80/tables
```

SQL Anywhere Studio インストール環境の *Samples¥ASA¥HTTP* サブディレクトリには、より多くの例が含まれています。

変数の使用

HTTP 要求の変数は、2 種類のソースのいずれかによって指定されます。最初の方法は、さまざまな名前 - 値の対が含まれたクエリを URL で指定することです。HTTP GET 要求はこのような形でフォーマットされます。

2 番目は、URL パスを介した方法です。URL PATH を ON または ELEMENTS に設定すると、パスの部分と後続のサービス名が変数値として解釈されます。このオプションにより、データベース内に格納されている何かを示す代わりに、従来のファイルベースの Web サイトのように URL が特定のディレクトリ内のファイルを要求するように指定できます。

たとえば、URL **http://localhost/gallery/sunset.jpg** は、gallery という名前のディレクトリから *sunset.jpg* というファイルを要求するように見えますが、そうではなくデータベース・テーブルから画像を検索するよう gallery サービスに要求します。

HTTP 要求で渡されるパラメータは、URL PATH の設定によって決まります。

- **OFF** サービス名の後にパス・パラメータを許可しません。
- **ON** サービス名の後のすべてのパス要素が、変数 URL に割り当てられます。

- **ELEMENTS** URL パスの残りの部分をスラッシュ文字で区切り、最大で 10 要素をリストできます。これらの値は、変数 URL1、URL2、URL3、…、URL10 と割り当てられます。値が 10 個より少ない場合、残りの変数は NULL に設定されます。11 個以上の変数を指定するとエラーになります。

定義されたロケーション以外は、変数に違いはありません。すべての HTTP 変数に同じようにアクセスし、使用します。たとえば、url1 などの変数値は、**?picture=sunset.jpg** のようなクエリの一部として指定されるパラメータと同じようにアクセスされます。

パラメータへのアクセス

変数にアクセスする主な方法がいくつかあります。1 つは、サービス宣言の文にある変数を使用する方法です。たとえば、次の文は複数の変数値を show_table ストアド・プロシージャに渡します。

```
CREATE SERVICE show_table TYPE 'RAW'
 AUTHORIZATION ON
 AS CALL show_table( :user_name, :table_name, :limit,
:start )
```

他の方法としては、要求を処理するストアド・プロシージャ内で組み込み関数 next_http_variable と http_variable を使用することです。どの変数が定義されているかが分からない場合は、next_http_variable を使用して検索できます。http_variable 関数によって変数値が返されます。

next_http_variable 関数により、定義された変数の名前で繰り返すことができます。最初にこれを呼び出すときには、NULL 値を渡します。すると、この関数が 1 つの変数名を返します。次にこれを呼び出すときから、前の変数の名前を渡すたびに、次の変数名を返すようになります。最後の変数名がこの関数に渡されると、NULL を返します。

この方法で変数名を繰り返し渡す場合、各変数名が正確に 1 回だけ返されることになります。ただし、変数が返される順番は、要求で指定された順番と同じでない場合もあります。さらに、これを繰り返した場合、2 回目は違う順番で返されます。

各変数の値を取得するには、http_variable 関数を使用します。最初のパラメータが変数の名前です。追加のパラメータはオプションです。1 つの変数に対して複数の値が指定される場合、1 つのパラメータのみが指定されると関数は最初の値を返します。2 番目のパラメータに整数を指定すると、追加の値を検索できます。

3 番目のパラメータで、変数ヘッダフィールド値をマルチパート要求から検索できます。ヘッダ・フィールド名を指定してこの値を検索します。たとえば、次の SQL 文は 3 つの変数値を検索し、次にイメージ変数のヘッダフィールド値を検索します。

```
SET v_id    = http_variable( 'id' );
 SET v_title = http_variable( 'title' );
 SET v_descr = http_variable( 'descr' );

SET v_name = http_variable( 'image', NULL, 'Content-
Disposition' );
 SET v_type = http_variable( 'image', NULL, 'Content-
Type' );
SET v_image = http_variable( 'image', NULL, '@BINARY');
```

自動文字セット変換

デフォルトで、文字セット変換はテキスト・タイプの出力結果セットで自動的に実行されます。バイナリ・オブジェクトなどの他のタイプの結果セットでは変換されません。要求の文字セットはデータベースの文字セットに変換され、必要に応じて結果セットはデータベースの文字セットからクライアントの文字セットに変換されます。結果セットのバイナリ・カラムは変換されません。要求に処理可能な文字セットが複数リストされている場合、サーバがリストの中から最初に検出した最適なものを使用します。

文字セット変換は、HTTP オプションの CharsetConversion を設定することで有効または無効にできます。使用できる値は ON と OFF です。デフォルト値は ON です。次の文は、文字セットの変換をオフにします。

```
call dbo.sa_set_http_option( 'CharsetConversion', 'OFF')
```

組み込みのストアド・プロシージャの詳細については、『ASA SQL リファレンス・マニュアル』>「システム・プロシージャとカタログ・ストアド・プロシージャ」を参照してください。

エラー

Web サービス要求に失敗した場合、データベース・サーバが標準エラーを生成してブラウザに表示します。これらのエラーには、プロトコル標準と一貫性のある番号が割り当てられています。

サービスが SOAP サービスである場合、SOAP バージョン 1.1 標準で定義されたように、フォールトは SOAP クライアントに対して SOAP フォールトとして返されます。

- 要求を処理するアプリケーションのエラーによって SQLCODE が生成されると、クライアントの **faultcode** により SOAP フォールトが返されます。その場合 Client.Procedure などのサブカテゴリが含まれる可能性もあります。SOAP フォールト内の **faultstring** 要素は、エラーの詳しい説明に設定されており、**detail** 要素には、数値の SQLCODE 値が含まれます。
- 転送プロトコル・エラーが発生した場合、**faultcode** は、そのエラーに応じて **Client** か **Server** のいずれかに設定されます。faultstring は `404 Not Found` などの HTTP 転送メッセージに設定され、**detail** 要素には数値の HTTP エラー値が含まれます。
- SQLCODE 値を返すアプリケーション・エラーのために生成された SOAP フォールト・メッセージは、`200 OK` という HTTP ステータスで返されます。

クライアントを SOAP クライアントとして識別できない場合は、生成された HTML ドキュメントで適切な HTTP エラーが返されます。

発生する可能性のある一般的なエラーは次のとおりです。

番号	名前	SOAP フォールト	説明
301	Moved permanently	Server	要求されたページは永続的に移動されました。サーバは、自動的に新しいロケーションに要求をリダイレクトします。

番号	名前	SOAP フォールト	説明
304	Not Modified	Server	サーバは、要求の情報に基づき、要求されたデータは前回の要求の後変更されているため、再度送信する必要はないと判断しました
307	Temporary Redirect	Server	要求されたページは移動されましたが、この変更は永続的なものではない可能性があります。サーバは、自動的に新しいロケーションに要求をリダイレクトします。
400	Bad Request	Client.BadRequest	HTTP 要求が正しくないか不正です
401	Authorization Required	Client.Authorization	サービスを使用するのに認証が必要ですが、有効なユーザ名とパスワードが入力されていません
403	Forbidden	Client.Forbidden	データベースにアクセスするパーミッションがありません
404	Not Found	Client.NotFound	指定したデータベースがサーバで実行されていないか、指定した Web サービスが存在しません
408	Request Timeout	Server.RequestTimeout	要求の受信中に最大接続アイドル時間が超過しました
411	HTTP Length Required	Client.LengthRequired	サーバは、クライアントが要求に Content-Length の指定を含めることを必要とします。通常、このエラーはデータをサーバにアップロードしているときに発生します。
413	Entity Too Large	Server	要求が最大許可サイズを超過しました
414	URI Too Large	Server	URI の長さが最大長を超過しました
500	Internal Server Error	Server	内部エラーが発生しました。要求が処理できませんでした。
501	Not Implemented	Server	HTTP 要求メソッドが GET、HEAD、または POST ではありません

番号	名前	SOAP フォールト	説明
502	Bad Gateway	Server	要求されたドキュメントがサードパーティのサーバにあり、サーバがサードパーティのサーバからエラーを受け取りました
503	Service Unavailable	Server	接続数が最大数を超過しました

第 2 部　データベース・ファイルの処理

この項では、データベース・ファイルのインストール、使用、バックアップの各方法について説明します。また、国際言語と文字セットの使用方法についても説明します。

第 8 章

ファイル・ロケーションとインストール設定

この章の内容

この章では、Adaptive Server Anywhere で使用するインストール環境とオペレーティング・システムの設定について説明します。これらの設定は、オペレーティング・システムに応じて、環境変数、初期化ファイル・エントリ、またはレジストリ設定として保管できます。

インストール・ディレクトリ構造

Adaptive Server Anywhere をインストールするとき、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、Adaptive Server 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。Adaptive Server Anywhere の付属ツールは、他のディレクトリにインストールされます。この項では、Adaptive Server 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。

Adaptive Server Anywhere インストール・ディレクトリ

Adaptive Server 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。

- **サンプル・データベース** サンプル・データベースは、ファイル *asademo.db* にあります。
- **『はじめにお読みください』** *readme.txt* ファイルに最新情報が記述されています。

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* または *win64* 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、これらのファイルは *bin* および *lib* ディレクトリにインストールされます。NetWare では、実行プログラムはインストール・ディレクトリに保存されます。

  使用するマシン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**　Java ベース・クラスはこのディレクトリに保存されます。
- **スクリプト・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。*custom.sql* スクリプトを除いて、*これらのスクリプトは編集しないでください*。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。
- **サンプル・ディレクトリ**　これは、各サンプル用に別のディレクトリを保持します。
- **h ディレクトリ**　*h* ディレクトリには、ESQL と ODBC データベース開発用のヘッダ・ファイルが保存されています。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、ファイルはすべてインストール・ディレクトリ *¥Program Files¥Sybase¥ASA9* にインストールされます。サブディレクトリは作成されません。ただし、DLL はすべて *¥Windows* ディレクトリにインストールされます。

Adaptive Server 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。

- Adaptive Server Anywhere を実行するには、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 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。

Adaptive Server 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。

広範囲なファイル検索

データベース・サーバと管理ユーティリティを含む Adaptive Server 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、Adaptive Server Anywhere プログラムは次の順番でファイルを検索します。

1. **実行ディレクトリ** 実行プログラム・ファイルが保存されています。

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があります。

   - 実行ディレクトリの親
   - 親ディレクトリの子 *scripts*。UNIX サーバはこのロケーションを検索しません。

3. **現在作業中のディレクトリ**　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、Adaptive Server 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* という名前の子
   - オペレーティング・システム名の付いた子 (*win32*、*win64* 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。

8. **ライブラリ・パス (UNIX のみ)**　UNIX オペレーティング・システムでは、ライブラリ・パスを設定する必要があります。設定が必要な対応する環境変数は、以下のいずれかになります。

   - LD_LIBRARY_PATH (Linux、Solaris、DEC)
   - DYLD_LIBRARY_PATH (Mac OS X)
   - SHLIB_PATH (HP-UX)

- LIBPATH (AIX)

環境変数

Adaptive Server Anywhere では、一連の環境変数を使用してさまざまなタイプの情報を保管します。この項では、環境変数を列挙して説明します。設定しなければならない変数は、状況によって異なります。

環境変数の設定

環境変数の設定方法は、使用しているオペレーティング・システムによって異なります。

❖ 環境変数を設定するには、次の手順に従います (Windows NT の場合)。

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 [プロパティ] を選択します。

2　[環境] タブをクリックします。環境変数がない場合、変数とその値を入力し、[設定] をクリックします。

変数が存在する場合、[システム環境変数] または [ユーザ環境変数] から変数を選択し、[値] フィールドを必要に応じて変更します。[設定] をクリックします。

❖ 環境変数を設定するには、次の手順に従います (Windows 2000 の場合)。

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 [プロパティ] を選択します。

2　[詳細] タブをクリックします。

3　[環境変数] ボタンをクリックします。

[環境変数] ダイアログが開きます。環境変数が存在しない場合は、[新規] をクリックし、変数とその値を適切な場所に入力し、[OK] をクリックします。

変数が存在する場合は、[システム環境変数] または [ユーザー環境変数] から変数を選択し、[編集] をクリックし、[変数値] フィールドを変更します。[OK] をクリックして、設定を変更します。

❖ 環境変数を設定するには、次の手順に従います (Windows XP の場合)。

1 [スタート] メニューで [設定] ー [コントロール パネル] ー [システム] を選択します。

2 [詳細] タブをクリックします。

3 [環境変数] ボタンをクリックします。

[環境変数] ダイアログが開きます。環境変数が存在しない場合は、[新規] をクリックし、変数とその値を適切な場所に入力し、[OK] をクリックします。

変数が存在する場合は、[システム環境変数] または [ユーザー環境変数] から変数を選択し、[編集] をクリックし、[変数値] フィールドを変更します。[OK] をクリックして、設定を変更します。

❖ 環境変数を設定するには、次の手順に従います (UNIX の場合)。

- スタートアップ・ファイル (*.cshrc*、*.shrc*、*.login*) のいずれかに、変数を設定する 1 行を追加します。

一部のシェル (sh、bash、ksh など) では、次のような行になります。

```
export VARIABLE=value
```

その他のシェル (csh、tsch など) では、次のような行になります。

```
setenv VARIABLE value
```

ASANY9 環境変数

構文　**ASANY9**=*directory-name*

デフォルト　*C:¥Program Files¥Sybase¥SQL Anywhere 9*

説明　ASANY9 環境変数は、インストール時に SQL Anywhere ディレクトリに設定されます。

この環境変数は必ず設定してください。設定が必要な理由はいくつかありますが、その 1 つは、サンプルが SQL Anywhere アプリケーションを探すときに、この環境変数が使用されるということがあります。

ASANYSH9 環境変数

構文　**ASANYSH9**=*directory-name*

デフォルト　なし

説明　この変数は、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するときに、Interactive SQL、Sybase Central、Adaptive Server Anywhere コンソール・ユーティリティ (dbconsole) によって使用されます。ASANYSH9 は、共有ディレクトリのロケーションに設定されます。インストール時に共有ディレクトリのロケーションを設定してください。デフォルトの共有ディレクトリは、*C:¥Program Files¥Sybase¥Shared* です。

ASCHARSET 環境変数

構文　**ASCHARSET**=*charset*

説明　*Charset* は文字セット名です。たとえば、**ASCHARSET=cp1252** の場合、デフォルトの文字セットが cp1252 に設定されます。

次の方法で最初に値を返すものが、デフォルトの文字セットを特定します。

- ASCHARSET 環境変数のチェック

- オペレーティング・システムのクエリ

文字セット情報が何も指定されていない場合、UNIX には iso_1、それ以外には cp850 を使用します。

この環境変数は、Windows CE または NetWare ではサポートされていません。

ASLANG 環境変数

構文 **ASLANG**=*language_code*

説明 *Language_code* は、言語を表す 2 文字の組み合わせです。たとえば、**ASLANG=DE** の場合、デフォルトの言語をドイツ語に設定します。

次の方法で最初に値を返すものが、デフォルトの言語を特定します。

- ASLANG 環境変数のチェック
- インストーラまたは *dblang.exe* で設定されたレジストリ・チェック (Windows のみ)
- オペレーティング・システムでの言語情報のクエリ

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。

この環境変数は、Windows CE または NetWare ではサポートされていません。

ASLOGDIR 環境変数

構文 **ASLOGDIR**=*directory-name*

説明 ASOGDIR 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

Windows では、ASLOGDIR が設定されていないと、サーバはサーバ実行プログラムが通常格納されているディレクトリで *backup.syb* ファイルを探そうとします。たとえば、32 ビットの Windows プラットフォームでは、*C:¥Program Files¥Sybase¥SQL Anywhere 9¥win32* がそのディレクトリに相当します。インストール・ディレクトリは、*HKLM¥Software¥Sybase¥Adaptive Server Anywhere¥9.0¥Location* レジストリ・キーを使用して決定されます。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。Location レジストリ・キーが存在しない場合は、サーバ実行プログラムに使用されるパスが指定されていればそれが調べられます。サーバを起動するためのパスが指定されていない場合、*backup.syb* ファイルは現在のドライブのルート・ディレクトリに保存されます。

UNIX

UNIX では、ASLOGDIR 環境変数が設定されていない場合、サーバは HOME 環境変数をチェックします。HOME が設定されている場合、*backup.syb* ファイルはそのディレクトリに保存されます。それ以外の場合、サーバはサーバを起動したユーザの Home ディレクトリにファイルを保存しようとします。そのディレクトリを判別できない場合、サーバはサーバが起動されたディレクトリに *backup.syb* ファイルを保存しようとします。

参照

- 『ASA SQL リファレンス・マニュアル』> 「BACKUP 文」

ASTMP 環境変数

構文

ASTMP=*directory-name*

デフォルト

なし

説明

データベース・サーバは ASTMP 環境変数の値をチェックして、テンポラリ・ファイルの保管されている場所を判断します。ASTMP 環境変数がない場合は、最初に存在する TMP、TMPDIR、TEMP 環境変数が使用されます。UNIX では、上記のどの環境変数も存在しない場合は、*/tmp* が使用されます。

ほとんどの場合、ASTMP は必要ありません。この変数は、データベース・サーバをサービスとして実行する場合に、セキュリティを考慮した環境で使用できます。これによって、他のプログラムがアクセスできないディレクトリにテンポラリ・ファイルを保存できます。

UNIX では、共有メモリ・アクセスで使用したファイルが Adaptive Server Anywhere のテンポラリ・ファイルに含まれています。アプリケーションが共有メモリを使用してデータベース・サーバに接続している場合、Adaptive Server Anywhere テンポラリ・ファイルが置かれているディレクトリに ASTMP 環境変数を設定する必要があります。

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。

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」374 ページを参照してください。

参照

- 「TEMP 環境変数」371 ページ

LD_LIBRARY_PATH 環境変数 [UNIX]

構文 **LD_LIBRARY_PATH**=*installation_path*/lib

説明

LD_LIBRARY_PATH 環境変数は UNIX でのみ使用されます。この環境変数は、Adaptive Server Anywhere ライブラリがあるディレクトリを含むよう、インストール・プログラムによって修正されます。

ライブラリは、インストール・ディレクトリの *lib* サブディレクトリにあります (たとえば、*/opt/SYBSasa9/lib*)。

Mac OS X では、DYLD_LIBRARY_PATH 環境変数が使用され、AIX では LIBPATH 環境変数が使用されます。

PATH 環境変数

構文 **PATH**=*installation_path*

説明

PATH 環境変数は、Adaptive Server Anywhere 実行プログラムのあるディレクトリを含むよう、インストール・プログラムによって修正されます。

実行プログラムはインストール・ディレクトリのサブディレクトリにあります。

他の Sybase アプリケーションを使用している場合、*SYBASE¥bin* と *SYBASE¥dll* ディレクトリがパスに追加されます。

UNIX では、各ユーザのパスに、実行プログラムのあるディレクトリ (*/opt/SYBSasa9/bin*) を追加します。

SQLCONNECT 環境変数

構文　**SQLCONNECT**=*parameter*=*value*; …

説明

SQLCONNECT 環境変数はオプションです。インストール・プログラムでは設定を行いません。

SQLCONNECT では、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。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。

SQLCONNECT 環境変数内の接続パラメータ文字列を設定するときは、等号の代わりにシャープ記号 (#) を使用します。環境変数設定の中で **=** を使うと構文エラーになります。

接続パラメータについては、「接続パラメータ」236 ページを参照してください。

SQLLOCALE 環境変数

SQLLOCALE 環境変数はサポートされなくなりました。ASLANG と ASCHARSET 環境変数に置き換わっています。

ASLANG 環境変数の詳細については、「ASLANG 環境変数」366 ページを参照してください。

ASCHARSET 環境変数の詳細については、「ASCHARSET 環境変数」365 ページを参照してください。

SQLPATH 環境変数

構文 **SQLPATH**=*path*; …

説明 SQLPATH 環境変数はオプションです。インストール・プログラムでは設定を行いません。

Interactive SQL は、システム・パスを検索する前に、SQLPATH に従ってコマンド・ファイルとヘルプ・ファイルを検索します。

SQLREMOTE 環境変数

構文 **SQLREMOTE**=*path*

説明 SQLREMOTE 環境変数はオプションです。インストール・プログラムでは設定を行いません。

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。この変数は共有ディレクトリを示します。

32 ビットの Windows オペレーティング・システムでは、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します。

SYBASE 環境変数

構文 **SYBASE**=*path*

説明 SYBASE 変数は、Adaptive Server Enterprise と *DSEdit* などのユーティリティを含む、Sybase アプリケーションのインストール用ホーム・ディレクトリを示します。Adaptive Server ファミリ製品と一緒に Adaptive Server Anywhere を使用するときにのみ、この変数が必要となります。

TEMP 環境変数

構文

ASTMP=*path*

TMP=*path*

TMPDIR=*path*

TEMP=*path*

説明

データベース・サーバは、ユニオンのソートや実行などの、さまざまな操作に対するテンポラリ・ファイルを作成します。テンポラリ・ファイルは、ASTMP、TMP、TMPDIR、または TEMP 環境変数によって指定されたディレクトリに作成されます。Adaptive Server Anywhere はこれらのうちの最初に検出されたものから使用します。

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、上記のどの環境変数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」374 ページを参照してください。

ASTMP は、データベース・サーバ (dbeng9 または dbsrv9) によってのみ使用されます。テンポラリ・ディレクトリを必要とするコマンド・ライン・ユーティリティは、サポートされるすべてのプラットフォームで TMP、TMPDIR、TEMP のみを使用します。

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、Adaptive Server 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によって行われるので、通常はユーザがレジストリ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。

Adaptive Server 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。

詳細については、「.ini ファイルの内容の非表示」679 ページを参照してください。

警告

Adaptive Server 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に単純暗号化を追加しないようにしてください。その他のデータ・ソース (Mobile Link 同期など) を使用する場合は、*.odbc.ini* ファイルの内容を読みにくくすると、その他のドライバが正しく機能しなくなる場合があります。

現在のユーザとローカル・マシン設定

Windows NT などのオペレーティング・システムの中には、2 レベルのシステム設定があるものがあります。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、マシン全体に関連し、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、マシンの管理者パーミッションを入手してください。

Adaptive Server 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 NT 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・マシン設定に優先します。

ローカル・マシン設定が必要なとき

Adaptive Server 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、マシン全体を停止しないかぎり、マシンを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。

Adaptive Server 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 PowerDynamo 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。Adaptive Server 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
         Adaptive Server Anywhere
             9.0
         Sybase Central
             4.3
             Profiles
             Providers
```

インストール時のレジストリ設定

インストール・プログラムは、Sybase レジストリ中で次のレジストリ設定を行います。

- **Location** *Adaptive Server Anywhere¥9.0* レジストリ・キーでは、インストール・ディレクトリのロケーションを入力します。次に例を示します。

  ```
  Location  "c:¥Program Files¥Sybase¥SQL Anywhere 9"
  ```

- **Language** *Adaptive Server Anywhere¥9.0* レジストリ・キーでは、メッセージとエラーで使用される現在の言語を示す 2 文字コードを入力します。次に例を示します。

  ```
  Language "EN"
  ```

  デフォルト設定は英語 (EN) です。インストール・プログラムは、ソフトウェアが英語以外の言語でインストールされたときにかぎり、このエントリを設定します。

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、レジストリで次のように設定します。

```
HKEY_CURRENT_USER¥Software¥Sybase¥Adaptive Server
Anywhere¥9.0¥TempFolder
```

TempFolder に、使用するテンポラリ・ディレクトリ名を指定します。サーバの動作は次のいずれかになります。

- 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、次のように処理されます。

- ディレクトリ *¥Temp* が存在する場合、そのディレクトリを使用する。
- ディレクトリ *¥Temp* が存在しない場合、ディレクトリ ¥Temp を作成する。

ディレクトリ *¥Temp* がなく、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 9 章

データベース・ファイルの処理

この章の内容

この章では、データベース・ファイルと関連ファイルの作成方法と処理方法について説明します。

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。

  データベースの作成については、『ASA SQL ユーザーズ・ガイド』>「データベースの編集」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、リカバリとレプリケーションに必要です。通常、このファイルの拡張子は *.log* です。

  トランザクション・ログについては、「トランザクション・ログ」491 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・ファイルを無視します。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・ファイルの格納先は、環境変数によって決定されます。

  以下の環境変数が順にチェックされます。

  - ASTMP
  - TMP
  - TMPDIR
  - TEMP

  これらがいずれも定義されていない場合、Adaptive Server Anywhere は、テンポラリ・ファイルを Windows オペレーティング・システムの現在のディレクトリ、または UNIX の */tmp/.SQLAnywhere* ディレクトリに格納します。

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。

詳細については、『ASA SQL リファレンス・マニュアル』>「sa_disk_free_space システム・プロシージャ」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。

- **追加データベース・ファイル**　データを複数の個別ファイルに分散できます。これらの追加ファイルを DB 領域と呼びます。

  DB 領域については、『ASA SQL リファレンス・マニュアル』>「CREATE DBSPACE 文」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。

  ミラーリングされたトランザクション・ログについては、「トランザクション・ログ・ミラー」492 ページを参照してください。

追加 DB 領域の使用

この項では、DB 領域と呼ばれる追加データベース・ファイルの使用方法について説明します。

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。

各データベース・ファイルの最大容量は、256 M 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 (256 M*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・サイズを最大 2 GB に制限していますが、Windows NT/2000/XP のファイル・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。

既存のデータベースの分割

既存のデータベース・オブジェクトを複数の DB 領域に分割する場合は、データベースをアンロードし、データベース再構築用に生成されたコマンド・ファイルを修正してください。これには、メイン・ファイルに配置しないテーブルごとに、IN 句を追加して DB 領域を指定します。

詳細については、『ASA SQL リファレンス・マニュアル』>「UNLOAD TABLE 文」を参照してください。

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。

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には、CREATE TABLE 文で IN 句を指定すると、特定の DB 領域に配置できます。IN 句を指定しない場合は、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、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] ー [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、新しい DB 領域 **library** 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
```

参照

- 『ASA SQL リファレンス・マニュアル』> 「CREATE DBSPACE 文」
- 『ASA SQL ユーザーズ・ガイド』> 「テーブルの作成」
- 『ASA SQL リファレンス・マニュアル』> 「CREATE INDEX 文」

データベース・ファイル用領域の事前割り付け

Adaptive Server Anywhere では、必要に応じてデータベース・ファイルの容量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。頻繁に変更されるデータベースでなければ、データベース・ファイルに対して明示的に領域を割り付ける必要はありません。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、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」381 ページ

- 『ASA SQL リファレンス・マニュアル』> 「ALTER DBSPACE 文」

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

- 『ASA SQL ユーザーズ・ガイド』> 「テーブルの削除」
- 『ASA SQL リファレンス・マニュアル』> 「DROP 文」

ライト・ファイルの処理 (旧式)

推奨されなくなった機能
ライト・ファイルは推奨されなくなりました。

読み込み専用データベース・ファイルがある場合 (たとえば、データベースを CD-ROM で配布する場合)、ライト・ファイルを使用してローカルでデータベースを変更できます。

ライト・ファイルを作成するには、ライト・ファイル作成ユーティリティ [dbwrite] または CREATE WRITEFILE 文を使用します。この項の例では、ユーティリティを使用しています。

CREATE WRITEFILE 文の詳細については、『ASA SQL リファレンス・マニュアル』> 「CREATE WRITEFILE 文 (旧式)」を参照してください。

ローカルで変更できないようにデータベースを読み込み専用で開く方法については、「-r サーバ・オプション」206 ページを参照してください。

❖ ライト・ファイルを使用するには、次の手順に従います。

1. データベースに対してライト・ファイルを作成します。

   たとえば、サンプル・データベースに対してライト・ファイルを作成するには、サンプル・データベース・ファイル *asademo.db* のコピーが格納されているディレクトリで、次のコマンドを実行します。

   ```
   dbwrite -c asademo.db
   ```

   このコマンドによって、*asademo.wrt* という名前のライト・ファイルと、*asademo.wlg* という名前のトランザクション・ログが作成されます。

2 データベース・サーバを起動し、ライト・ファイルをロードします。デフォルトでは、サーバはまず拡張子 *.wrt* の付いたファイルを検索するため、次のコマンドを使用すると、サンプル・データベースのライト・ファイルを実行するパーソナル・サーバが起動されます。

```
dbeng9 asademo
```

サーバ・ウィンドウのメッセージが、起動したファイルを示します。

3 Interactive SQL を使用してデータベースに接続します。サンプル・データベースがデフォルトなので、ユーザ ID **DBA** とパスワード **SQL** を使用できます。

4 通常どおりにクエリを実行します。たとえば、次のクエリを使用して department テーブルの内容をリストします。

```
SELECT *
 FROM department
```

department テーブルのデータが、データベース・ファイル *asademo.db* から取得されます。

5 ローを挿入します。次の文を使用して、department テーブルにローを 1 つ挿入します。

```
INSERT
 INTO department (dept_id, dept_name)
 VALUES (202, 'Eastern Sales')
```

この変更をコミットした場合、変更はトランザクション・ログ *asademo.wlg* に書き込まれ、データベースのチェックポイントが実行された場合、ライト・ファイル *asademo.wrt* に書き込まれます。

ここで department テーブルに問い合せると、ライト・ファイルとデータベース・ファイルから結果が取り出されます。

6 ローを削除します。ここでは参照整合性の問題を避けるため、WAIT_FOR_COMMIT オプションを設定します。

```
SET TEMPORARY OPTION wait_for_commit = 'on' ;
 DELETE
 FROM department
 WHERE dept_id = 100
```

この変更を行った場合は、ライト・ファイル内で削除のマークが付けられます。データベース・ファイルは変更されません。

目的によっては、共有データベースでライト・ファイルを使用すると便利です。たとえば、ネットワーク・サーバ上に読み込み専用ファイルがあれば、各ユーザはライト・ファイルを個別に持つことができます。こうすると、ユーザはローカルの情報を追加し、共有データベースに影響を与えずに自分のマシンに保存できます。この方法はアプリケーション開発にも役立ちます。

ライト・ファイルの削除

dberase ユーティリティを使用すると、ライト・ファイルとその関連トランザクション・ログを削除できます。

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。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。

ユーティリティ・データベースの機能によって、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。

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'
```

これらの文の構文の詳細については、『ASA SQL リファレンス・マニュアル』>「CREATE DATABASE 文」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' )
```

接続プロパティとサーバ・プロパティのリストについては、「データベース・プロパティ」923 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

ユーティリティ・データベースに接続しているときに使用できる文は、次に示すものだけです。

- ALTER DATABASE *dbfile* MODIFY TRANSACTION LOG
- CREATE DATABASE
- CREATE DECRYPTED FILE
- CREATE ENCRYPTED FILE
- DROP DATABASE

- RESTORE DATABASE
- 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。ユーザ ID とパスワードの要件は、パーソナル・サーバとネットワーク・サーバとで異なります。

パーソナル・データベース・サーバの場合、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・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。

詳細については、「ユーティリティ・データベースのパスワード」391 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。

   ```
   dbeng9.exe -n TestEng
   ```

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、実際のデータベースをロードしないで、**TestEng** という名前のパーソナル・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・サーバの場合、接続に関するセキュリティ上の制限があります。パスワードは、データベース・サーバ実行プログラムと同じディレクトリにあるファイル *util_db.ini* に格納されます。

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。

.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」679 ページを参照してください。

❖ 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 データベース・サーバ実行プログラムと同じディレクトリにあるファイル *util_db.ini* に、パスワードを追加します。たとえば、次の *util_db.ini* ファイルのパスワードは ASA です。

```
[UTILITY_DB]
 PWD=ASA
```

2 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。

```
dbsrv9.exe -n TestEng
```

3 Interactive SQL を起動します。

4 [接続] ダイアログで、ユーザ ID に **DBA** と入力し、ファイル *util_db.ini* に格納されているパスワードを入力します。

5 [データベース] タブで、データベース名に **utility_db** と入力します。

6　[OK] をクリックして接続します。

Interactive SQL は、実際のデータベースをロードしないで、**TestEng** という名前のネットワーク・サーバ上のユーティリティ・データベースに接続します。

詳細については、「データベースへの接続」49 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・サーバのセキュリティ

ユーティリティ・データベース・サーバのセキュリティには、2 つの側面があります。

- ユーティリティ・データベースに接続できるユーザの制限。これは、パスワードを使用することによって制御します。
- ファイル管理文を実行できるユーザの制限。これは、データベース・サーバのオプションによって制御します。

ユーティリティ・データベースのパスワード

パーソナル・サーバとネットワーク・サーバとでは、接続用のセキュリティ・モデルが異なります。

パーソナル・サーバの場合、ユーザ ID **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・サーバは単一マシンでの利用を目的としているため、パスワードなどのセキュリティ上の制限は必要ありません。

ネットワーク・サーバの場合、ユーザ ID **DBA** を指定します。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。

util_db.ini ファイル　*util_db.ini* ファイルの内容は、次のとおりです。

```
[UTILITY_DB]
 PWD=password
```

utility_db セキュリティ・レベルの使用は、データベース・サーバをホスト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めてしまうからです。

util_db.ini ファイルの内容を保護するためには、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で内容を読みにくくします。

.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」679 ページを参照してください。

ファイル管理文の実行パーミッション

セキュリティ・レベルを使用することによって、特定の管理タスクを実行できます。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。

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。それらのレベルには、**all**、**none**、**DBA**、および **utility_db** があります。**utility_db** レベルは、ユーティリティ・データベースへの接続権限を持つ人にだけ、ファイル管理文の使用を許可します。

-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

データベース・サーバの -gu オプションの詳細については、「-gu サーバ・オプション」197 ページを参照してください。

例

- ファイル管理文を使用させないようにするには、-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、データベース・サーバ名は **TestSrv** になります。サンプル・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。

  ```
  dbsrv9.exe -n TestSrv -gu none asademo.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。

  ```
  dbsrv9 -n TestSrv -gu utility_db
  ```

  ユーティリティ・データベースのパスワードが、インストール中 **asa** に設定されている場合は、次のコマンドを使って、Interactive SQL ユーティリティをクライアント・アプリケーションとして起動してサーバ **TestSrv** に接続し、ユーティリティ・データベースをロードしてユーザを接続します。

  ```
  dbisql -c
  "uid=DBA;pwd=asa;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 10 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

この章の内容

この章では、Adaptive Server Anywhere のスケジュール機能とイベント処理機能を使用して、管理タスクやその他の作業を自動化する方法について説明します。

概要

データベース管理タスクの多くは、体系的に行うと効果的です。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 Adaptive Server 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。

章の内容

この章では、次の情報について説明します。

- スケジュールとイベント処理の概要 (この項)
- スケジュールとイベント・ハンドラの設計、使用に役立つ概念と補足情報
  - 「スケジュールの概要」399 ページ
  - 「システム・イベントの概要」401 ページ
- イベント・ハンドラを開発する方法について
  - 「イベント・ハンドラの開発」406 ページ
- 内部についての情報
  - 「スケジュールとイベントの内部」408 ページ
- 自動化したタスクの実行方法の段階的な説明
  - 「イベント処理タスク」411 ページ

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」399 ページ
イベントとは？	「イベントの概要」398 ページ
システム・イベントとは？	「システム・イベントの概要」401 ページ
イベント・ハンドラとは？	「イベント・ハンドラの概要」406 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」406 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」409 ページ
定期バックアップをスケジュールする方法は？	例については、「スケジュールの概要」399 ページを参照してください。
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」401 ページ 『ASA SQL リファレンス・マニュアル』>「CREATE EVENT 文」
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」410 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」406 ページ 『ASA SQL リファレンス・マニュアル』>「EVENT_PARAMETER 関数 [システム]」

イベントの概要

イベントには 3 種類あります。

- 「**スケジュールされたイベント**」は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。
- 「**システム・イベント**」は、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。
- 「**手動イベント**」は、TRIGGER EVENT 文を使用して明示的に起動されます。

各イベント・タイプについては、以下の項で詳しく説明します。

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。

次に便利なスケジュールの例を示します。

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。

詳細については、『ASA SQL リファレンス・マニュアル』>「CREATE EVENT 文」を参照してください。

例

毎日、午前 1 時にインクリメンタル・バックアップを実行します。

```
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 DBA.OrderSummary
        SELECT current date,
                 count( * ),
                 sum( amount )
        FROM DBA.Orders
        WHERE date_ordered = current date
END
```

スケジュールの定義

スケジュール定義にはいくつかの構成要素があり、柔軟に設定することができます。

- **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、イベントが最初に実行される時刻です。
- **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。

システム・イベントの概要

データベース・サーバは、何種類かのシステム・イベントを追跡します。システム・イベントがデータベース・サーバによって確認され、設定した「**トリガ条件**」を満たすと、イベント・ハンドラがトリガされます。

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。

使用可能なシステム・イベントの詳細については、「システム・イベントの選択」401 ページを参照してください。

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」403 ページを参照してください。

システム・イベントの選択

Adaptive Server Anywhere は、いくつかのシステム・イベントを追跡しています。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、(イベント・ハンドラに定義された) アクションを必要なときに実行します。

使用可能なシステム・イベントは次のとおりです。

- **バックアップ**　**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。
- **データベース起動**　**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。
- **切断**　**切断**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・イベントは次のオペレーティング・システムでは使用できません。

  - OSR2 より前の Windows 95
  - Windows CE

  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 オプションを指定すると、ファイル・システム・フルの状態が発生したときにコールバック関数を実行できます。

  詳細については、「-fc サーバ・オプション」186 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。event_condition 関数と RemainingValues をこのイベント・タイプの引数として使用できます。

- **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。

イベントの作成については、『ASA SQL リファレンス・マニュアル』>「CREATE EVENT 文」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、Adaptive Server 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。

詳細については、『ASA SQL リファレンス・マニュアル』> 「EVENT_PARAMETER 関数 [システム]」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ほうが便利です。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。

詳細については、『ASA SQL リファレンス・マニュアル』> 「TRIGGER EVENT 文」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。

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。

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。

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、『ASA SQL リファレンス・マニュアル』> 「EVENT_PARAMETER 関数 [システム]」を参照してください。

段階を追った手順の詳細については、「イベント・ハンドラのデバッグ」414 ページ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。

実行に 65 分かかるイベント・ハンドラが 9 時 ～ 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 ～ 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。

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 でも、SQL でも同じような方法で処理されます。

詳細については、「スケジュールの概要」399 ページと「システム・イベントの概要」401 ページを参照してください。

❖ 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] フォルダをクリックします。
3. [ファイル] メニューから [新規] ー [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。

詳細については、『ASA SQL リファレンス・マニュアル』>「CREATE EVENT 文」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] フォルダをクリックします。
3. [ファイル] メニューから [新規] ー [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. イベント・ハンドラに対する SQL 文を入力し、[次へ] をクリックします。
7. [このイベントを有効にする] と [すべてのロケーションで実行] を選択してから、[次へ] をクリックします。
8. 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。

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

- イベントのトリガについては、「イベント・ハンドラのトリガ」413 ページを参照してください。
- イベントの変更については、『ASA SQL リファレンス・マニュアル』> 「ALTER EVENT 文」を参照してください。

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・ハンドラの開発の詳細については、「イベント・ハンドラの開発」406 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右クリックし、ポップアップ・メニューから [トリガ] を選択します (イベントを有効にしてから、ポップアップ・メニューの [トリガ] を選択してください。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
```

詳細については、『ASA SQL リファレンス・マニュアル』> 「TRIGGER EVENT 文」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・ハンドラの開発については、「イベント・ハンドラの開発」406 ページを参照してください。

デバッガの使用については、『ASA SQL ユーザーズ・ガイド』> 「データベースでのデバッグ論理」を参照してください。

❖ 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] メニューから、[プログラム] ー [SQL Anywhere 9] ー [Sybase Central] を選択します。

2　データベースに接続します。

3　デバッグ・モードに変更します。

[タスク] メニューから [デバッグ] を選択します。[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名をクリックします。

6　[SQL] ウィンドウ枠で、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。

第 11 章

国際言語と文字セット

この章の内容

この章では、Adaptive Server 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説明します。

国際言語と文字セットの概要

この章では、複数の文字セットを使用する環境で作業を行う場合、または英語以外の言語を使用する場合に直面する可能性のある問題についての概要を説明します。

データベースを作成するときは、使用するデータベースの照合順 (照合ともいう) を指定します。照合とは、データベース内の文字に対する文字セットとソート順の組み合わせです。

Adaptive Server Anywhere のローカライズ版

Adaptive Server Anywhere には、次の言語について完全ローカライズ版 (パッケージ、ソフトウェア、マニュアル、ブック) があります。

- 英語
- ドイツ語
- フランス語
- 日本語
- 中国語 (簡体文字)

同様に、Adaptive Server Anywhere には、次の言語展開版 (ソフトウェアのみ) があります。

- 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

Adaptive Server Anywhere の国際化機能

Adaptive Server Anywhere には、データベースの言語を設定する場合に特に関係がある、2 つの機能セットがあります。

- **照合**　データベースの作成時に、幅広い選択肢の中から照合を選択できます。適切な照合を使用してデータベースを作成すると、データは正しくソートされます。

  データベースが、文字列の比較、ソート、または大文字と小文字の変換などの文字列操作を実行するときは、必ず照合順を使って行います。データベースがソートと文字列の比較を実行する文を次に示します。

  - ORDER BY 句の入ったクエリ
  - LOCATE、SIMILAR、SOUNDEX などの文字関数を使う式
  - LIKE を使った条件
  - 文字データに対してインデックス・ルックアップを使用するクエリ

  データベースは、照合を使用して、カラム名などの有効な識別子やユニークな識別子を調べます。

- **文字セット変換**　サーバとクライアント・システムの文字セット・コード間でデータを変換するように、Adaptive Server Anywhere を設定できます。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。

  文字セットの変換は、クライアントとサーバとの間で行われます。また、Adaptive Server Anywhere ODBC ドライバでも行われます。OLE DB と ADO.NET でもこの機能が提供されており、すべてユニコードに対応しています。

デフォルトの照合の使い方

デフォルトのアクションを使用してデータベースを作成すると、初期化ユーティリティは、データベースが作成されたマシンのオペレーティング・システムが使用している文字セットと言語から照合を決定します。

自分の環境のデフォルト照合を検索する方法については、「デフォルトの照合の検索」458 ページを参照してください。

使用している環境にあるすべてのマシンが同じ文字セットを共有する場合は、デフォルト照合によって文字セット変換の必要性が最小化されます。選択された照合の文字セットとソート順が使用するデータに適しているかぎり、このデフォルト設定を使用すれば、最も簡単にシステム全体で一貫して文字を表示できます。

システムにデフォルト設定を使用できない場合は、データベースで使用する照合を決定する必要があります。また、文字セット変換を使用して、データベース・システム内で一貫してデータを交換できるようにするかどうかも決定してください。この章では、このような決定とその実行に必要な情報について説明します。

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 …
文字セットを正しく処理できるようにコンピューティング環境を設定する方法は？	「文字セット環境の設定」458 ページ
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」438 ページ
ソフトウェア、特に Adaptive Server Anywhere における文字の表示方法は？	「ソフトウェアの文字セットの知識」422 ページ
Adaptive Server Anywhere が提供する照合の種類は？	「照合の選択」438 ページ

質問 …	参照先 …
データベース・サーバからクライアント・アプリケーションへ送信されるエラー・メッセージや情報メッセージが、使用するアプリケーションに対して適切な言語や文字セットであることを確認する方法は？	「データベース・メッセージの文字変換」448 ページ
クライアント・マシンで使用している文字セットとデータベースで使用している文字セットが違う。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？	「文字セット変換を使用してデータベース・サーバを起動する」464 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」448 ページ
提供されたものと異なる照合を作成する方法は？	「カスタム照合を使用してデータベースを作成する」467 ページ
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」468 ページ
Windows CE 用のデータベースを作成する方法は？	「Windows CE 用データベースの作成」445 ページ

ソフトウェアの文字セットの知識

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。

文字セットの問題

コンピュータ・ソフトウェアによる文字記憶領域と表示には、次のようないくつかの異なる面があります。

- 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。
- これらの文字を処理するために、各ソフトウェアは文字セットの「**コード**」を採用します。各文字は、一般的に 16 進数として表示される 1 バイトまたは複数バイトの情報にマッピングされます。このコードは「**コード・ページ**」とも呼ばれます。
- 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 16 進数間のマップ) と文字の「**ソート順**」の組み合わせです。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、言語間で文字の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、文字セットの順序が想定されます。

  データベース・サーバはクライアント・アプリケーションからバイトのストリームとして文字列を受け取ります。また、データベースの文字セットに従って、これらのバイトと文字を関連付けます。データがインデックス付けされたカラムに保持される場合、そのインデックスは照合のソート順に従ってソートされます。

- **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 Adaptive Server Anywhere が呼び出す 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・オペレーティング・システムは、インタフェースにテキストを表示します。テキストの処理も実行できます。

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。

コード・ページ

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、Code Page 1253 はギリシャ文字のセット、Code Page 1252 はさまざまな言語の文字を表すのに適した国際的な文字セットです。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、Microsoft 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 ～ 127 はすべてのシングルバイト・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、大文字小文字の英語のアルファベット、共通の句読表記記号と数字が含まれます。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 ～ 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 ～ 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。

例

フランス語の文字列を保持するデータベースで Code Page 850 を使用していて、クライアント・オペレーティング・システムで Code Page 437 を使用するとします。文字 Ã (大文字 A 抑音) は、文字 ¥xB7 (10 進数値 183) としてデータベースに保持されます。Code Page 437 では、文字 ¥xB7 は図形文字です。クライアント・アプリケーションがこのバイトを受け取り、オペレーティング・システムがそれを画面に表示します。ユーザに対して A 抑音ではなく図形文字が表示されます。

Windows 環境の ANSI コード・ページと OEM コード・ページ

ほとんどの PC では少なくとも 2 つのコード・ページが使用されているため、PC ユーザにとっては問題は複雑です。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。これらのコード・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 95/98/Me と Windows NT/2000/XP 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。

Adaptive Server 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。

例

次の状況を考えてみます。

- PC が、ANSI Code Page 1252 を使って Windows オペレーティング・システムを実行している。
- 文字モード・アプリケーションのコード・ページは、OEM Code Page 437。
- テキストは、UTF8 を使用して作成されたデータベースに保持されている。

大文字 A 抑音は、データベースに 16 進バイト C380 として保存されます。Windows アプリケーションでは同じ文字が 16 進数 CO として、DOS アプリケーションでは 16 進数 B7 として表されます。

Adaptive Server Anywhere では、文字セット変換によりこれらの表現の違いを処理しています。

データベースの照合の選択の詳細については、「照合の知識」438 ページを参照してください。

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、マルチバイトの文字セットを使います。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。

マルチバイト文字セットには 2 種類あります。「**可変幅**」のものは、いくつかの文字はシングルバイトで、他はダブルバイトなどになります。「**固定幅**」のものは、すべての文字が同じバイト数になります。

マルチバイト文字セットの詳細については、「マルチバイト照合の使い方」446 ページを参照してください。

例

たとえば、シフト JIS 文字セットの文字は 1 バイトまたは 2 バイトの長さを持ちます。最初のバイトの値が、16 進数 ¥x81 ～ ¥x9F または ¥xE0 ～ ¥xEF (10 進数値 129 ～ 159 または 224 ～ 239) の範囲にある場合、その文字は 2 バイト文字であり、フォロー・バイトと呼ばれる後続のバイトで文字が完成されます。「**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。

最初のバイトがこの範囲外にある場合、その文字は 1 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。

- シフト JIS 文字では、文字の種類を最初のバイトから読みとることができます。最初のバイトが ¥x09 ～ ¥x0D、または ¥x20 である文字はスペースです。
- 文字が ¥x41 ～ ¥x5A、¥x61 ～ ¥x7A、¥x81 ～ ¥x9F、¥xE0 ～ ¥xEF のいずれかの範囲にある場合、アルファベット (文字) であると考えられます。
- 文字が ¥x30 ～ ¥x39 の範囲にある場合は、アラビア数字です。

照合を使用した文字のソート

データベースの照合順は文字のアルファベット順での考え方を含み、それを数字やスペースを含む、文字セットのすべての文字に拡張しています。

各ソート位置に複数の文字を関連付ける

各ソート位置には複数の文字を割り当てられます。これは、たとえばアクセント付きの文字をアクセントなしの文字と同等に扱いたいときなどに便利です。

ソート位置が同じ 2 つの文字はデータベースではまったく同じものとして扱われます。したがって、照合で *a* と *e* が同じソート位置に割り当てられていると、探索条件を持つクエリは次のようになります。

```
WHERE col1 = 'want'
```

探索条件を持つクエリは、**col1** カラムのエントリ **went** があるローによって満たされます。

ソート位置ごとに大文字と小文字を指定できます。大文字と小文字が区別されるデータベースでは、大文字と小文字は同じものとしては扱われません。区別されないデータベースでは、大文字と小文字は同じものとして扱われます。

マルチバイト文字に対する最初のバイトの照合順

マルチバイト文字のソート順は、最初のバイトにのみ設定できます。最初のバイトの値が同じ文字は、それ以降のバイトの 16 進値によってソートされます。

外国語での大文字と小文字の区別

Adaptive Server Anywhere では通常、テーブル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**」。つまり、名前は作成時の大文字と小文字を区別して格納されますが、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・テーブルの名前は大文字 (SYSDOMAIN、SYSTABLE 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、上記の 2 つの文は同等です。

```
SELECT *
 FROM systable

 SELECT *
 FROM SYSTABLE
```

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。特に、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** という文字を含むシステム・オブジェクトまたは **I** 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

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。

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、トルコ語では次の文字が小文字の ı に相当するものです。

ı

一方 ANSI では次のようになります。

```
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、次の文について考えます。

```
SELECT * FROM sysinfo    // actual table name is SYSINFO
SELECT * FROM fig        // actual table name is FİG
```

最初の文はシステム・オブジェクトを参照し、ANSI SQL の規則を使用して名前を一致させますが、2 番目の文はユーザ・オブジェクトを参照し、トルコ語の規則を使用します。

しかし、オブジェクトが検索されるまで、データベース・サーバがどの規則を使用するかを判断するコンテキストがありません。つまり、オブジェクトを検索するには、大文字と小文字の変換規則を認識する必要があります。そしてこの規則を認識するには、オブジェクトを検索する必要があるのです。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。

この場合、Adaptive Server Anywhere はトルコ語の規則を使用します。最初の文は失敗しますが、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 SYSINFO
```

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 sysınfo // 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。Adaptive Server 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 Adaptive Server Anywhere は、組み込み関数などの特定の識別子にこのやり方を適用します。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 規則を優先したほうがうまくいく場合もあります。

以前のバージョンの Adaptive Server Anywhere では、カスタム照合を作成し I-dot と I-no-dot を同等の文字とすることでこれを可能にしていました。Adaptive Server 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。

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・ライブラリによって使用され、どの文字セットに結果を返すか決定します。文字セット変換が有効な場合 (デフォルト)、データベース・サーバは自身のロケールとアプリケーションのロケールを比較して、文字セット変換が必要かどうか判断します。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。

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、ロケール言語を使用して次の動作を決定します。

  - どの言語ライブラリをロードするか。
  - 照合が明示的に指定されていない場合、データベース作成時に文字セットと共に言語を使用して、照合を決定する。

  クライアント・ライブラリは、ロケール言語を使用して次の動作を決定します。

  - どの言語ライブラリをロードするか。
  - どの言語をデータベースから要求するか。

    詳細については、「ロケール言語の知識」434 ページを参照してください。

- **文字セット** 使用しているコード・ページです。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、相互運用性を有効にするために文字セット変換が必要です。

  OEM コード・ページと ANSI コード・ページの両方を使用するマシンの場合、ここでは ANSI コード・ページの値を使用します。

  詳細については、「ロケール文字セットの知識」436 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を示します。

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」460 ページを参照してください。

ロケールの言語コンポーネントは、クライアント・ライブラリによって次の手順で決定されます。

1. Windows では、Adaptive Server Anywhere の言語レジストリ・エントリを確認します。これについては、「インストール時のレジストリ設定」374 ページに記述されています。

2. その他のオペレーティング・システムの場合、またはレジストリ設定が存在しない場合は、オペレーティング・システムの言語設定を確認します。

ロケールの言語コンポーネントは、データベース・サーバによって次の手順で決定されます。

1. ASLANG 環境変数が存在する場合は、これを確認します。

   詳細については、「ASLANG 環境変数」366 ページを参照してください。

2. Windows では、Adaptive Server Anywhere の言語レジストリ・エントリを確認します。これについては、「インストール時のレジストリ設定」374 ページに記述されています。

3. オペレーティング・システムを問い合わせます。

4. 上記の設定で言語を決められない場合、デフォルトの英語になります。

言語ラベルの値

次の表に、有効な言語ラベルの値と対応する ISO 639 ラベルを示します。

言語ラベル	代替ラベル	ISO_639 言語コード
chinese	simpchin	ZH
czech	なし	CS
danish	なし	DA
dutch	なし	NL
finnish	なし	FI
french	なし	FR
german	なし	DE
greek	なし	EL
hebrew	なし	HE
hungarian	なし	HU
italian	なし	IT
japanese	なし	JA
korean	なし	KO
lithuanian	なし	LT
norwegian	norweg	NO
polish	なし	PL

言語ラベル	代替ラベル	ISO_639 言語コード
portuguese	portugue	PT
russian	なし	RU
spanish	なし	ES
swedish	なし	SV
tchinese	tradchin	TW
turkish	なし	TR
ukrainian	なし	UK
us_english	english	EN

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、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。文字セット変換が使用可能な場合 (デフォルト)、データベース・サーバは自身の文字セットとアプリケーションの文字セットを比較して、文字セット変換が必要かどうか判断します。

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」460 ページを参照してください。

文字セットは、クライアント・ライブラリによって次の手順で決定されます。

1. 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]」241 ページ を参照してください。

2. Open Client アプリケーションは、Sybase *locales* ディレクトリの *locales.dat* ファイルを確認します。

3. 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・システムでは、Windows ANSI 文字セットです。
   - UNIX では、デフォルトで ISO8859-1 になります。
   - その他のプラットフォームでは、Code Page 1252 を使用します。

接続用文字セットは、データベース・サーバによって次の手順で決定されます。

1. クライアントによって指定された文字セットがサポートされている場合は、それを使用します。

   詳細については、「CharSet 接続パラメータ [CS]」241 ページ を参照してください。

2. クライアントによって指定された文字列がサポートされていない場合は、データベースの文字セットを使用します。

照合の知識

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」462 ページと「初期化ユーティリティ」687 ページを参照してください。

データベースの照合の変更については、「データベースの照合を変更する」468 ページを参照してください。

照合の選択

データベースを作成すると、Adaptive Server Anywhere によってオペレーティング・システムの言語と文字セットの設定に基づいたデフォルト照合が選択されます。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。

Adaptive Server Anywhere が複数の照合をサポートする場合もあります。照合は、その中に含まれている文字と、合字 (2 つ以上の文字を結合した文字) やアクセント付き文字などの特殊文字のソート方法によって区別されます。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。

データのソートと国際化機能の詳細については、「Adaptive Server Anywhere の国際化機能」419 ページを参照してください。

Adaptive Server Anywhere での新しいデータベースの照合の選択

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、Adaptive Server Anywhere では言語と文字セットを使用して照合が決定されます。

- 言語は、ASLANG 環境変数 (存在する場合) またはオペレーティング・システムによって決まります。
- 文字セットは、ASCHARSET 環境変数 (存在する場合) またはオペレーティング・システムによって決まります。

最後の手段として選択される照合
Adaptive Server Anywhere がオペレーティング・システムの文字セットと言語に基づいて特定の照合を決定できなかった場合は、使用されているオペレーティング・システムに関係なく、Adaptive Server Anywhere レジストリ設定または ASCHARSET 環境変数 ISO_BINENG がデフォルトになります。

この設定が要件に適していない場合は、データベースを再構築して、別の照合を選択してください。

照合の変更の詳細については、「データベースの照合を変更する」468 ページを参照してください。

特定の照合を使用したデータベースの作成については、「名前を付けた照合を使用してデータベースを作成する」462 ページを参照してください。

提供されている照合と推奨する照合

次の照合が Adaptive Server Anywhere で提供されます。コマンド・プロンプトで次のコマンドを入力すると、主な照合のリストが表示されます。

```
dbinit -l
```

照合ラベル	説明
874THAIBIN	Code Page 874、Windows タイ語
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)

照合ラベル	説明
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
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
1254TRK	Code Page 1254 Windows ラテン語 5、トルコ語、ISO 8859-9 拡張付き
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_JAPAN	日本語の EUC JIS X 0208-1990 と JIS X 0212-1990 コード
EUC_CHINA	中国語 (簡体文字) の GB 2312-80 コード
EUC_TAIWAN	台湾語の Big 5 コード
EUC_KOREA	韓国語の KS C 5601-1992 コード、Johab (組合型)
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

照合ラベル	説明
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
UTF8	UTF-8 (ユニコード変換フォーマット -8)、ユニコード用の 8 ビット・マルチバイト・コード化

おすすめするのは次の照合です。これらは、アプリケーションが使用している文字セットと一致する可能性が高い、つまり、ソート順序が適切で、特定の言語で必要とされる記号とアクセント付き文字がサポートされているからです。おすすめする UNIX 照合がない言語もあります。

言語	Windows での照合	UNIX での照合
西ヨーロッパ言語 (英語、フランス語、ドイツ語、イタリア語、ポルトガル語、スペイン語など)	1252LATIN1	ISO9LATIN1、ISO1LATIN1
東ヨーロッパ言語 (クロアチア語、チェコ語、ハンガリー語、ルーマニア語、セルビア語、スロバキア語、スロベニア語など)	1250LATIN2	
アラビア語	1256ARA	
ギリシャ語	1253ELL	
ヘブライ語	1255HEB	
日本語	932JPN	EUC_JAPAN
韓国語	949KOR	EUC_KOREA
リトアニア語	1257LIT	
ポーランド語	1250POL	

言語	Windows での照合	UNIX での照合
ロシア語とウクライナ語	EUC_CHINA	
中国語 (簡体文字)	936ZHO	EUC_CHINA
スペイン語	1252SPA	ISOLATIN1、ISO9LATIN1
タイ語	874THAIBIN	874THAIBIN
中国語 (繁体文字)	950ZHO_TW	
中国語 (繁体文字) + HKSCS	950ZHO_HK	
トルコ語	1254TRK	

ヒント
ドイツ語用の照合を選択するコードでは、1252DEU *ではなく*、1252LATIN1 を選択してください。1252DEU ではウムラウト記号の有無によって文字が区別されますが、1252LATIN1 では区別されません。1252LATIN1 では Muller と Mullller は同じものとみなされますが、1252DEU ではこの 2 つは同じものとみなされません。1252DEU ではウムラウト付きの文字が別の文字として認識されるので、アルファベットでの文字の順序は "ob, oa" のようになります。

代替照合

代替照合は、古いバージョンの Adaptive Server Anywhere との互換性や、その他の特殊な用途で使用できます。コマンド・プロンプトで次のコマンドを入力すると、代替照合のリストが表示されます。

```
dbinit -l+
```

-l+ オプションを指定すると、次の照合のリストがおすすめする照合のリストに追加されます。

照合ラベル	タイプ	説明
437LATIN1	OEM	Code Page 437、ラテン語 1、西ヨーロッパ言語
437ESP	OEM	Code Page 437、スペイン語
437SVE	OEM	Code Page 437、スウェーデン／フィンランド語
819CYR	ANSI	Code Page 819、キリル語
819DAN	ANSI	Code Page 819、デンマーク語
819ELL	ANSI	Code Page 819、ギリシャ語
819ESP	ANSI	Code Page 819、スペイン語
819ISL	ANSI	Code Page 819、アイスランド語
819LATIN1	ANSI	Code Page 819、ラテン語 1、西ヨーロッパ言語
819LATIN2	ANSI	Code Page 819、ラテン語 2、中央／東ヨーロッパ言語
819NOR	ANSI	Code Page 819、ノルウェー語
819RUS	ANSI	Code Page 819、ロシア語
819SVE	ANSI	Code Page 819、スウェーデン／フィンランド語
819TRK	ANSI	Code Page 819、トルコ語
850CYR	OEM	Code Page 850、キリル語、西ヨーロッパ言語
850DAN	OEM	Code Page 850、デンマーク語
850ELL	OEM	Code Page 850、ギリシャ語
850ESP	OEM	Code Page 850、スペイン語
850ISL	OEM	Code Page 850、アイスランド語

照合ラベル	タイプ	説明
850LATIN1	OEM	Code Page 850、ラテン語 1、西ヨーロッパ言語
850LATIN2	OEM	Code Page 850、ラテン語 2、中央／東ヨーロッパ言語
850NOR	OEM	Code Page 850、ノルウェー語
850RUS	OEM	Code Page 850、ロシア語
850SVE	OEM	Code Page 850、スウェーデン／フィンランド語
850TRK	OEM	Code Page 850、トルコ語
852LATIN2	OEM	Code Page 852、ラテン語 2、中央／東ヨーロッパ言語
852CYR	OEM	Code Page 852、キリル語
852POL	OEM	Code Page 852、ポーランド語
855CYR	OEM	Code Page 855、キリル語
856HEB	OEM	Code Page 856、ヘブライ語
857TRK	OEM	Code Page 857、トルコ語
860LATIN1	OEM	Code Page 860、ラテン語 1、西ヨーロッパ言語
861ISL	OEM	Code Page 861、アイスランド語
862HEB	OEM	Code Page 862、ヘブライ語
863LATIN1	OEM	Code Page 863、ラテン語 1、西ヨーロッパ言語
865NOR	OEM	Code Page 865、ノルウェー語
866RUS	OEM	Code Page 866、ロシア語
869ELL	OEM	Code Page 869、ギリシャ語
920TRK	ANSI	Code Page 920、トルコ語、ISO-8859-9

照合ラベル	タイプ	説明
1252DEU	ANSI	Code Page 1251、Windows 特殊ドイツ語、ウムラウト記号の有無による区別あり
1254TRKALT	ANSI	Code Page 1254、Windows 特殊トルコ語、I-dot と I-no-dot の区別なし 詳細については、「トルコ語文字セットと照合」428 ページを参照してください。

Windows CE 用データベースの作成

Windows CE は、ユニコードに基づくオペレーティング・システムです。Adaptive Server Anywhere ODBC ドライバは、ASCII (8 ビット) またはユニコード文字列のいずれかをサポートし、必要に応じて文字セット変換を実行します。Embedded SQL アプリケーションの開発をしている場合は、Windows API 関数を使用して、データベースからユニコード・バージョンの文字列を取得できます。

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。

ANSI 照合の注意事項

ISO_1 照合

ISO_1 は、Adaptive Server Enterprise のデフォルト ISO_1 照合と互換性があります。相違点は次のとおりです。

- 小文字の sharp s (¥xDF) は、Adaptive Server Anywhere では小文字 **s** でソートしますが、Adaptive Server Enterprise では **ss** の後でソートします。

- **AE** と **ae** (¥xC6 と ¥xE6) に対応する合字は、Adaptive Server Anywhere ではそれぞれ **A** と **a** の後でソートされますが、Adaptive Server Enterprise では **AE** と **ae** の後でソートされます。

1252LATIN1 照合

この照合には、ユーロ通貨記号と他のいくつかの文字 (Z-with-caron と z-with-caron) が含まれています。この照合は、英語または西ヨーロッパ言語を使用する Windows ユーザにおすすめします。

ユーロ記号は、その他の通貨記号でソートします。

ISO1LATIN1 照合

この照合は ISO_1 と同じですが、A0 ～ BF の範囲の値のソートで使用されます。Adaptive Server Enterprise との互換性を保つため、ISO_1 照合には 0xA0 ～ 0xBF に対する文字がありません。ただし、ISO ラテン語 1 の文字セットでは、これらの場所に文字があります。ISO1LATIN1 照合は、これらの場所の文字を反映します。

Adaptive Server Enterprise との互換性が必要でない場合、通常は ISO_1 の代わりに ISO1LATIN1 を使用することをおすすめします。

ISO1LATIN1 は、英語または西ヨーロッパ言語を使用する UNIX ユーザにおすすめします。

ISO9LATIN1 照合

この照合は ISO1LATIN1 と同じですが、ユーロ通貨記号とその他の ISO1LATIN1 照合の新しい文字が含まれています。

ISO ラテン語 9 の文字セットを使用する場合は、この照合を選択してください。

マルチバイト照合の使い方

この項では、マルチバイト文字セットを処理する方法について説明します。以下の説明は、サポートされている照合と、ユーザが作成するマルチバイトのカスタム照合に適用されます。

Adaptive Server Anywhere は、複数のマルチバイト文字セットの照合を提供します。

詳細については、「照合の知識」438 ページを参照してください。

Adaptive Server Anywhere は、8 ビット文字セットをサポートします。この文字セットでは、ある文字は 1 バイトで表され、他の文字は最長 4 バイトまでの複数バイトで表されます。文字の最初のバイトは、その文字を使用するバイト数とその文字の種類 (スペース、数字、またはアルファベット) を示します。

UTF8 照合の場合、UTF-8 文字は 1 から 4 バイトを表します。他のマルチバイト照合の場合、1 または 2 バイトが使用されます。提供されているすべてのマルチバイト照合において、2 バイト以上で構成されている文字は「アルファベット」と見なされ、二重引用符なしの識別子で使用できます。

Embedded SQL 以外のすべてのクライアント・ライブラリは、UTF-16 エンコードを使用したユニコードに対応しています。変換は、クライアントとサーバ間で行われます。

文字セット変換の知識

Adaptive Server Anywhere は、文字セットまたはコード・ページの異なる場所にあって同じ文字を表す文字セット間で、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 とシフト JIS の文字セット間では文字セット変換を実行できますが、EUC-JIS と OEM Code Page 850 の間では実行できません。

この項では、Adaptive Server Anywhere の文字セット変換方法について説明します。この情報は、複数の文字セットを使用する環境でアプリケーションやデータベースを展開するユーザなどの、上級ユーザ向けです。

データベース・メッセージの文字変換

データベース・ソフトウェアが出すエラー・メッセージとその他のメッセージは、言語リソース・ライブラリに保持されます。このライブラリのローカライズ版は、ローカライズ版の Adaptive Server Anywhere と共に提供されます。

クライアント・アプリケーションのユーザは、データベースのデータだけでなくメッセージも見ることができます。

メッセージは、文字セット変換の設定に関係なく、常にデータベース文字セットに変換されます。

データベース・サーバで文字セットの変換がオン (デフォルト) であり、クライアントの文字セットがデータベース照合で使用されている文字セットと異なる場合は、さらに文字セット変換が実行されます。

接続文字列と文字セット

接続文字列による特殊な場合の文字セット変換があります。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。解析はサーバの文字セットや言語を認識せずに実行されます。

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. *keyword* = *value* コンポーネントに分解されます。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。サーバ名は、クライアント・マシンの文字セットに従って解釈されます。Windows オペレーティング・システムでは、ANSI 文字セットが使用されます。クライアント・マシンとサーバ・マシンの間で文字セット変換の問題を引き起こさないかぎり、拡張文字を使用できます。

   異なるマシン間で最大の互換性を得るには、アルファベットまたは数値の ASCII 文字 0 ～ 127 (または 33 ～ 126) とアンダースコアだけを使用し、句読点文字は使用しないでください。サーバ名は 40 文字でトランケートされます。

3. DatabaseName (DBN) 接続パラメータまたは DatabaseFile (DBF) 接続パラメータは、データベース・サーバの文字セットで解釈されます。

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解釈され、データベースの文字セットに変換されます。

文字セット変換の回避

文字セット変換にはパフォーマンス・コストがかかります。文字セット変換の必要がない環境を設定できる場合は、このコストがかからず、管理も簡単です。Adaptive Server Anywhere では、クライアントの文字セットがサーバの文字セットと同じ場合は、文字セット変換は無効になります。

シングルバイト文字セット (SBCS) で作業をしていて、7 ビット ASCII 文字 (値 0 ～ 127) だけが関連する場合、文字セット変換は必要ありません。コード・ページがデータベース内やクライアントのオペレーティング・システムで異なっている場合でも、この範囲にわたって互換性があります。英語のインストール環境の多くは、この条件に当て

はまります。Adaptive Server Anywhere 9.0 以降では、文字セット変換はデフォルトでオンになっています。-ct- オプションを使用して、この設定をオフにできます。

詳細については、「データベース・サーバで文字セット変換をオフにする」464 ページを参照してください。

拡張文字を使用する必要がある場合は、次の手順に従います。

- クライアント・マシンのオペレーティング・システムのコード・ページが、データベース内で使用されているものと一致する場合は、データベース内のデータには文字セット変換は必要ありません。

  たとえば、多くの環境では 1252LATIN1 照合をデータベースで使用することが適切です。これは多くのシングルバイト環境の ANSI コード・ページに対応します。

- ユーザの言語で構築されたバージョンの Adaptive Server Anywhere を使用でき、オペレーティング・システムでそのコード・ページを使用する場合、データベース・メッセージには文字セット変換は必要ありません。Adaptive Server Anywhere のメッセージ文字列で使用される文字セットは、次のとおりです。

言語	Windows 文字セット
英語	cp1252
フランス語	cp1252
ドイツ語	cp1252
イタリア語	cp1252
日本語	cp932 (シフト JIS)
韓国語	cp949
リトアニア語	cp1257
ポーランド語	cp1250
ポルトガル語	cp1252

言語	Windows 文字セット
ロシア語	cp1251
中国語 (簡体文字)	cp936
スペイン語	cp1252
中国語 (繁体文字)	cp950
ウクライナ語	cp1251

また、クライアント／サーバでの文字セット変換は、デフォルトで実行されることを忘れないでください。-ct- オプションを使用して、この設定をオフにできます。

詳細については、「-ct サーバ・オプション」174 ページを参照してください。

内部照合

この項では、照合ファイルのファイル・フォーマットなどの、照合の内部技術の詳細について説明します。

この項は、カスタム照合を使用してデータベースを作成する場合に、特に役に立つ情報を提供します。この項で説明するファイル・フォーマットは照合の表示または編集に使用されます。その後、ファイルはデータベース・サーバで使用できるフォーマットに変換されます。

必要な手順については、「カスタム照合の作成」465 ページと「カスタム照合を使用してデータベースを作成する」467 ページを参照してください。

提供された照合とは異なる照合を使用してデータベースを作成できます。この項では、カスタム照合を使用してデータベースを構築する方法について説明します。

マルチバイトのカスタム照合を構築するときは、最初のバイトがどの範囲にあるとシングルバイトを示すか、ダブルバイト (または、それ以上) を示すか、スペース、アルファベット、数字を示すかを指定できます。

照合ファイルには次に示す要素が含まれています。

- コメント行 (データベースによって無視される)
- タイトル行
- 照合セクション
- コード・セクション
- プロパティ・セクション

コメント行

照合ファイルでは、スペースは通常無視されます。コメント行は、パーセント記号 **(%)** またはダッシュ 2 つ (–) のいずれかで始まります。

タイトル行

コメント行を除いた最初の行は次のフォームに従います。

Collation *collation_label (collation_name) (ase_charset) (ase_so_sensitive) (ase_so_insensitive) (java_charset)*

引数の説明

引数	説明
Collation	必須キーワード
collation_label	照合ラベル。システム・テーブル中に SYS.SYSCOLLATION.collation_label と SYS.SYSINFO.default_collation として表示される。このラベルは 10 文字以内で入力する。
collation_name	照合の説明。通常は文字セットと照合の順序について記述する。
ase_charset	この照合の文字セットと一致する文字セットの Adaptive Server Enterprise 名。この名前は、サーバ文字セット変換が有効になっているとき (デフォルト) に文字セットのマッピングを行うために使用される。
ase_so_sensitive	この照合と一致する、大文字と小文字を区別する Open Client または Adaptive Server Enterprise 照合の名前
ase_so_insensitive	この照合と一致する、大文字と小文字を区別しない Open Client または Adaptive Server Enterprise 照合の名前
java_charset	この照合の文字セットと一致する Java 文字セットの名前。この名前は、Java 仮想マシンとデータベースとの間で文字データを変換するときに使用される。

たとえば、932JPN 照合ファイルには、ラベル 932JPN と名前 (Code Page 932, Japanese Shift-JIS with Microsoft extensions) を持つ、次の照合行があります。

```
Collation 932JPN (Code Page 932, Japanese Shift-JIS
with Microsoft extensions) (cp932) (bin_cp932)
(bin_cp932) (SJIS)
```

照合セクション

タイトル行の後で、コメント行以外の各行は、照合内で 1 つの位置を記述します。行の順番はデータベースで使用されるソート順を決定し、また比較の結果を決定します。ファイル内でより上の部分 (始まりに近い部分) にある行の文字は後にある文字より先にソートされます。

各行のフォーマットを次に示します。

[*sort-position*] : *character* [[, *character*] …]

または

[*sort-position*] : *character* [*lowercase uppercase*]

引数の説明

引数	説明
sort-position	省略可。この行の文字をソートする位置を指定する。小さな数字は値が小さいことを示し、ソートではより始まりに近く置かれる。通常これは省略され、文字は 1 つ前の行にある文字に続く位置に置かれる。
character	ソート位置を指定する対象の文字
lowercase	省略可。この文字に対応する小文字を指定する。指定がなければ、対応する小文字はない。
uppercase	省略可。この文字に対応する大文字を指定する。指定がなければ、対応する大文字はない。

1 つの行にカンマで区切られた複数の文字が含まれることがあります。これらの文字は同一の文字としてソートされ、比較されます。

文字とソート位置の指定

各文字とソート位置を指定するフォーマットを次に示します。

指定	説明
¥d*nnn*	10 進数字で ¥d001 のように 0 ～ 9 を使う
¥x*hh*	16 進数字で ¥xB4 のように数字 0 ～ 9 と文字 a ～ f または A ～ F を使う

指定	説明
'c'	すべての文字 (',' など)
c	引用符 (')、円記号 (¥)、コロン (:)、カンマ (,) 以外の文字。これらの文字は以前のフォーマットを使う。

次に照合の行の例を示します。

```
% Sort some special characters at the beginning:
 : ' '
 : _
 : ¥xF2
 : ¥xEE
 : ¥xF0
 : -
 : ','
 : ;
 : ':'
 : !
 % Sort some letters in alphabetical order
 : A a A
 : a a A
 : B b B
 : b b B
 % Sort some E's from code page 850,
 % including some accented extended characters:
 : e e E, ¥x82 ¥x82 ¥x90, ¥x8A ¥x8A ¥xD4
 : E e E, ¥x90 ¥x82 ¥x90, ¥xD4 ¥x8A ¥xD4
```

その他の構文に関する注意

大文字と小文字を区別しないソートと比較を使用するデータベース (dbinit コマンドで -c が指定されていない) では、大文字と小文字のマッピングを使って、一緒にソートされる大文字と小文字を検索します。

マルチバイトの文字セットでは、最初のバイトが照合順にリストされ、最初のバイトが同じ文字は一緒にソートされ、それ以降のバイトの値に応じて並べられます。次にシフト JIS の照合ファイルの一部を例として示します。

```
: ¥xfb
 :  ¥xfc
 :  ¥xfd
```

この照合では、最初のバイトが ¥xfc であるすべての文字は、最初のバイトが ¥xfb であるすべての文字の後、¥xfd であるすべての文字の前にソートされます。2 バイト文字の ¥xfc ¥x01 は、同じように 2 バイト文字の ¥xfc ¥x02 よりも前にソートされます。

照合から省略された文字は、その照合の最後に追加されます。照合ファイルを処理するツールは警告を発します。

コード・セクション

コード・セクションはオプションで、照合順セクションに続きます。シングルバイト文字セットには必要ありません。

コード・セクションには、マルチバイト文字セットの最初のバイト値と、その有効なフォロー・バイトのバイト値がリストされます。

たとえば、シフト JIS コード・セクションは次のようになります。

```
Encodings:
 [¥x00-¥x80,¥xa0-¥xdf,¥xf0-¥xff]
 [¥x81-¥x9f,¥xe0-¥xef][¥x40-¥xff]
```

セクション・タイトルに続く最初の行は、有効なシングルバイト文字のリストです。角カッコ ([]) はカンマで区切られた範囲のリストを含みます。各範囲は値のペアをハイフンで区切ったものです。シフト JIS コードでは、¥x00 ～ ¥x80 の値は有効なシングルバイト文字ですが、¥x80 は有効ではありません。

セクション・タイトルに続く 2 行目は有効なマルチバイト文字をリストします。角カッコは範囲を示します。最初の範囲の 1 バイトとそれに続く 2 番目の範囲の 1 バイトの組み合わせは有効な文字です。

プロパティ・セクション

プロパティ・セクションはオプションで、コード・セクションに続きます。

プロパティ・セクションがある場合は、コード・セクションも必要です。

プロパティ・セクションには、アルファベット、数字、スペースの各文字の最初のバイト値がリストされています。

シフト JIS のプロパティ・セクションを次に示します。

```
Properties:
 space: [¥x09-¥x0d,¥x20]
 digit: [¥x30-¥x39]
 alpha: [¥x41-¥x5a,¥x61-¥x7a,¥x81-¥x9f,¥xe0-¥xef]
```

これは、最初のバイトが ¥x09 ～ ¥x0d の範囲と ¥x20 の文字はスペースであることを示します。数字は ¥x30 ～ ¥x39、アルファベットは ¥x41 ～ ¥x5a、¥x61 ～ ¥x7a、¥x81 ～ ¥x9f、¥xe0 ～ ¥xef の 4 つの範囲にあります。

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検索

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。

❖ **使用するマシンに対するデフォルトの照合を検索するには、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」438 ページを参照してください。

文字セット環境の設定

この項では、文字セットの問題を正しく処理できるように、使用するコンピューティング環境を設定する方法について説明します。ロケール環境を正しく設定すれば、使用するデータベースの照合を明示的に選択する必要はありません。クライアントとサーバ間の文字セット変換はオフにしておくことができます。

文字セット変換の無効化の詳細については、「データベース・サーバで文字セット変換をオフにする」464 ページを参照してください。

❖ 文字セット環境を設定するには、次の手順に従います。

1　使用する環境のコンピューティング・プラットフォームごとに、デフォルトのロケールを決定します。デフォルトのロケールとは、各コンピュータの文字セットと言語のことです。Windows オペレーティング・システムの文字セットは、ANSI コード・ページです。

ロケール情報の検索方法については、「ロケール情報の確認」460 ページを参照してください。

2　そのロケール設定が使用する環境に適しているかどうかを判断します。

詳細については、「照合の知識」438 ページを参照してください。

3　デフォルトの設定が適切でない場合は、文字セット、言語、データと一致し、文字セット変換を必要としないデータベース照合を決定します。

詳細については、「文字セット変換の回避」449 ページを参照してください。

4　その環境の各マシンのロケールに、これらの値を設定します。

詳細については、「ロケールの設定」461 ページを参照してください。

5　デフォルトの照合を使用してデータベースを作成します。デフォルトの照合では不十分な場合は、照合を指定してデータベースを作成します。

詳細については、「名前を付けた照合を使用してデータベースを作成する」462 ページと「データベースの照合を変更する」468 ページを参照してください。

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。

- 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よくあります。

- 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・コストもかかります。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、文字セット変換が自動的に無効になります。

  文字セット変換を回避するには、クライアント・マシン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と一致するデータベースの照合順を使用します。クライアント・マシン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。文字セット変換は、バージョン 8.0 以降の Adaptive Server Anywhere データベース・サーバではデフォルトで有効になっています。-ct- オプションを使用して、文字セット変換をオフにできます。

  詳細については、「文字セット変換の回避」449 ページを参照してください。

ロケール情報の確認

システム関数を使用して、ロケール情報を確認することができます。

詳細については、『ASA SQL リファレンス・マニュアル』> 「システム関数」を参照してください。

❖ データベース・サーバのロケールを確認するには、次の手順に従います。

1. Interactive SQL を起動し、データベース・サーバに接続します。
2. 次の文を実行して、データベース・サーバの文字セットを確認します。

   ```
   SELECT PROPERTY( 'CharSet' )
   ```

   「文字セット・ラベル」480 ページにリストされたサポートされている文字セットの 1 つが、クエリによって返されます。

3　次の文を実行して、データベース・サーバの言語を確認します。

```
SELECT PROPERTY( 'Language' )
```

「言語ラベルの値」435 ページにリストされたサポートされている言語の 1 つが、クエリによって返されます。

4　次の文を実行して、データベース・サーバのデフォルトの照合を確認します。

```
SELECT PROPERTY( 'DefaultCollation' )
```

「照合の選択」438 ページにリストされた照合の 1 つが、クエリによって返されます。

注意　クライアントのロケール情報を取得するには、クライアント・マシンで実行しているデータベース・サーバに接続します。

個々のデータベースに対する文字セットを取得するには、次の文を実行します。

```
SELECT DB_PROPERTY ( 'CharSet' )
```

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、明示的に Adaptive Server Anywhere コンポーネントが使用するロケールを設定することができます。

❖ コンピュータに Adaptive Server 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」460 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、次のように ASLANG と ASCHARSET 環境変数のいずれかまたはその両方を設定します。

```
ASCHARSET=charset;
ASLANG=language_code
```

ここで *charset* は、「文字セット・ラベル」480 ページのリストにある文字セット・ラベルです。*language_code* は、「言語ラベルの値」435 ページのリストにある言語ラベルです。

異なるオペレーティング・システムで環境変数を設定する方法については、「環境変数の設定」363 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。

❖ 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。

```
437LATIN1  Code Page 437, Latin 1, Western
 437ESP     Code Page 437, Spanish
 437SVE     Code Page 437, Swedish/Finnish
 819CYR     Code Page 819, Cyrillic
 819DAN     Code Page 819, Danish
 ...
```

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'
```

文字セット変換を使用してデータベース・サーバを起動する

文字セット変換は、クライアントとサーバのロケールが異なる場合に実行されます。文字セット変換は、バージョン 8.0 以降の Adaptive Server Anywhere データベース・サーバではデフォルトで有効になっています。バージョン 7.x 以前のデータベース・サーバでは、データベース・サーバのコマンドで文字セット変換を明示的に有効にしてください。

❖ **データベース・サーバで文字セット変換を有効にするには、次の手順に従います。**

- -ct+ オプションを使用して、データベース・サーバを起動します。次に例を示します。

```
dbsrv9 -ct+ asademo.db
```

データベース・サーバで文字セット変換をオフにする

文字セット変換は、デフォルトでは有効になっています。文字セット変換が必要ない場合は、データベース・サーバのコマンドで -ct- オプションを使用して無効にできます。

❖ **データベース・サーバで文字セット変換を無効にするには、次の手順に従います。**

- -ct- オプションを使用して、データベース・サーバを起動します。次に例を示します。

```
dbsrv9 -ct- asademo.db
```

詳細については、「-ct サーバ・オプション」174 ページを参照してください。

カスタム照合の作成

提供されている照合の中に要求を満たすものがない場合は、提供されている照合を修正して「**カスタム照合**」を作成できます。このカスタム照合を使用して、データベースを作成できます。

提供されている照合の詳細については、「照合の選択」438 ページを参照してください。

❖ カスタム照合を作成するには、次の手順に従います。

1. **開始照合を決める**　カスタム照合を作成するための開始点として、作成したい照合にできるだけ近い照合を選択してください。

   推奨する照合のリストについては、「照合の知識」438 ページを参照してください。

   または -l (小文字の L) オプションを指定して dbinit を実行します。

   ```
   dbinit -l
   ```

2. **カスタム照合ファイルを作成する**　照合 [dbcollat] ユーティリティを使用してこれを実行します。出力は照合ファイルです。

   たとえば、次の文は、1252LATIN1 照合を *mycustomcol.col* という名前のファイルに抽出します。

   ```
   dbcollat -z 1252LATIN1 mycustomcol.col
   ```

3. **カスタム照合ファイルを編集する**　テキスト・エディタで照合ファイル (この場合は *mycustomcol.col*) を開きます。

4 **照合ラベルを変更する** カスタム照合ファイルに自由に変更を加え、照合ラベルを定義します。たとえば、次のような Collation 行があり、1252LATIN1 照合のカスタム版を作成するとします。

```
Collation 1252LATIN1 (Code Page 1252, Windows Latin
1, Western) (cp1252) (dictionary_iso_1)
(nocase_iso_1) (Cp1252)
```

上記の行を以下のように変更します。

```
Collation MyOrdering (Code Page 1252, My Company
Ordering) (cp1252) (dictionary_iso_1) (nocase_iso_1)
(Cp1252)
```

この行の他のエントリは、Adaptive Server Anywhere 照合ラベルと、Java と Sybase TDS インタフェースがその照合情報に与えている名前を関連付けています。これらのインタフェースを使用していない場合は、これらのエントリを変更する必要はありません。

Collation 行は、次のようになります。

```
Collation collation_label (collation_name)
(ase_charset) (ase_so_sensitive)
(ase_so_insensitive) (java_charset)
```

文字セットのラベル (*ase_charset*) と 2 つのソート順ラベル (*ase_so_sensitive* と *ase_so_insensitive*) は、現在の照合に最も近い Open Client 文字セットとソート順を示します。*java_charset* ラベルは、Java が認識する最も近い文字セットです。

この行を編集して新しいラベルを付けてください。変更する必要があるラベルは *collation_label* です。手順 2 の例では、`1252LATIN1` です。

5 **照合定義を変更する** カスタム照合ファイルに自由に変更を加え、新しい照合を定義します。

照合ファイルの内容とフォーマットの詳細については、「内部照合」452 ページを参照してください。

6　**ファイルを SQL スクリプトに変換する**　dbcollat ユーティリティで -d オプションを指定して実行します。

たとえば、次のコマンドは *mycustomcol.col* 照合ファイルから *mycustom.SQL* ファイルを作成します。

```
dbcollat -d mycustomcol.col mycustom.SQL
```

7　**SQL スクリプトをインストール環境のスクリプトに追加する**　データベース作成時に使用されたスクリプトは、Adaptive Server Anywhere インストール・ディレクトリの scripts サブディレクトリに保持されています。*mycustom.SQL* の内容を、*custom.SQL* の末尾に追加してください。

これで新しい照合が作成され、データベース作成時に使用できるようになります。

カスタム照合を使用してデータベースを作成する

提供される照合の中に要求を満たすものがない場合は、カスタム照合を使用してデータベースを作成できます。カスタム照合はインデックスや文字列の比較に使用されます。

❖ カスタム照合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　カスタム照合を作成します。

データベース作成時に使用するカスタム照合が必要です。

カスタム照合の作成方法の詳細については、「カスタム照合の作成」465 ページを参照してください。

2　新しいデータベースを作成します。

初期化 [dbinit] ユーティリティを使用して、カスタム照合の名前を指定します。

たとえば、次のコマンドは、カスタム照合順 **newcol** を使用して *temp.db* という名前のデータベースを作成します。

```
dbinit -z newcol temp.db
```

Sybase Central で初期化ユーティリティを使用することもできます。

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、さまざまな理由があります。たとえば、次のような場合は、照合の変更が特に役立つことがあります。

- 設定環境全体で文字セット変換を回避する。
- データベース内の文字がデータベースの照合と一致しない。データベース内に定義されたものと同じ文字セットを使用することは、ソートをする場合に特に重要です。
- 別の文字セットを使用する。たとえば、OEM 文字セットから Windows 文字セットに切り替える場合があります。

既存のデータベース内の照合を単に変更するだけではそのデータベースのすべてのインデックスが無効になるので、そのような操作は許可されていません。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します。データベースを再構築すると、(照合設定などの) 新しい設定と再構築前のデータベースのデータを使用した新しいデータベースが作成されます。

データベースの照合を変更するときに考慮すべきケースが 2 つあります。この 2 つの違いは、データの文字セットを変更する必要があるかどうかです。

例 1

最初の例では、データベースのデータは照合に対して不適切な文字セットです。変更が必要なのは照合のみで、データの文字セットは変更しないでください。照合問題を解決するには、新しい照合設定と古いデータを使用して、データベースを再構築する必要があります。

英語環境において、古いデータベースで 850LATIN1 照合が使用されていたとします。このデータベースの中に Windows の「ウィンドウを使う」アプリケーションから挿入されたデータが含まれていた場合、そのデータは実際には CP1252 文字セットを使用している可能性があり、850LATIN1 照合で使用されている CP850 とは一致しません。

この状況は、ORDER BY 句がアクセント記号つきの文字を正しくソートしていないときによく見られます。この問題を修正するには、1252LATIN1 照合を使用して新しいデータベースを作成し、古いデータベースから新しいデータベースへ、変換を行わずにデータを移動します。データの変換を行わないのは、データがすでに新しいデータベースの照合と一致する文字セット (CP1252) になっているためです。

変換が行われないようにする最も簡単な方法は、-ct- オプションを指定してサーバを起動することです。次に、通常どおりデータベースを再構築します。

データベースの再構築については、『ASA SQL ユーザーズ・ガイド』>「データベースの再構築」を参照してください。

データベースの作成時の照合の指定については、「名前を付けた照合を使用してデータベースを作成する」462 ページを参照してください。

例 2

第 2 の状況では、照合と文字セットの両方を変更する必要があります。照合と文字セットの問題を解決するには、新しい設定を使用してデータベースを再構築してから、データの文字セットを変更する必要があります。

850LATIN1 データベースが、CP850 文字セットの文字を格納するように正しく使用されていたとします。しかし、文字セットの変換を避けるために、照合と文字セットの両方を更新したいとします。1252LATIN1 を使用して新しいデータベースを作成し、古いデータベースのデータを CP850 文字から CP1252 に変換しながら新しいデータベースに移します。

データベース内のデータの文字セットの変換は、サーバのクライアント／サーバ変換機能を使用して行われます。クライアント・アプリケーションとサーバ間の通信中に文字データを変換します。データベースの照合は、通信のデータベース・サーバ側の文字セットを決定します。また、オペレーティング・システムのロケール設定は、クライアントのデフォルト文字セットを決定します。ただし、クライアントの文字セットは、CharSet (CS) 接続パラメータによって上書きできます。

CharSet (CS) 接続パラメータの詳細については、「CharSet 接続パラメータ [CS]」241 ページ を参照してください。

文字セットの変換は、クライアント・アプリケーションとサーバとの通信中に発生するので、外部アンロードまたは再ロードが必要になります。内部アンロードと再ロードを使用した場合は、文字セットの変換は行われません。

❖ データベースの照合を変換し、データベースの文字セットを (変換または再ロードを使用して) 変更するには、次の手順に従います。

1 ソース・データベースからデータをアンロードします。

アンロード・ユーティリティを使用して、*reload.SQL* ファイルと一連のデータ・ファイルをソース・データベースの文字セットに作成できます。このフェーズ中はいかなる変換も実行したくないので、ソース・データベースを実行しているサーバで文字セット変換が有効になっていないことを確認します。バージョン 8 より前のバージョンでは、-ct が指定されていないことを確認します。バージョン 8 以降のサーバを使用している場合は、サーバの起動時に -ct- (文字セット変換なし) が指定されていることを確認します。

アンロード／再ロードが 1 台のマシンで行われる場合は、-ix オプションを使用して内部アンロードと外部再ロードを実行します。アンロード／再ロードが複数のマシン間で行われる場合は、-xx オプションを指定したうえでアンロード・ユーティリティを使用して、外部アンロードと外部再ロードを実行します。

「外部」アンロードまたは「外部」再ロードとは、アプリケーション (dbunload と dbisql ユーティリティ) がデータベースでカーソルを開き、ディスクにデータの読み込みまたは書き込みを行うことです。文字セット変換が行われます。「内部」アンロードまたは再ロードとは、UNLOAD TABLE または LOAD TABLE を使用して、サーバ自体がデータを読み込みまたは書き込みを行うことです。文字セット変換は*行われません*。

特定のテーブルのデータをアンロードする場合は、-t オプションまたは Interactive SQL の OUTPUT 文を使用してください。

アンロード・ユーティリティの詳細については、「アンロード・ユーティリティ」762 ページを参照してください。

2　適切な照合を使用するターゲット・データベースを作成します。初期化ユーティリティに -z オプションを含めて、ターゲット・データベースの照合順を指定します。

データベースの作成と再ロードには、そのデータベースの実行に使用するサーバと同じバージョンのサーバとツールを使用してください。

データベースの作成時の照合の指定については、「名前を付けた照合を使用してデータベースを作成する」462 ページを参照してください。

3　文字セット変換が有効になっているサーバでターゲット・データベースを起動します。-z オプションも指定できます。

–z オプションは必須ではありませんが、使用される文字セットと変換が行われる文字セットの検証が可能になります。サーバ・ウィンドウに各接続の文字セット変換設定が表示されます。ただし、–z サーバ・オプションを使用すると、再ロード中のパフォーマンスが低下する可能性があります。

詳細については、「文字セット変換を使用してデータベース・サーバを起動する」464 ページを参照してください。

4　手順 1 で *reload.SQL* ファイルを作成している場合は、次のようなコマンドを使用して、そのファイルをコマンド・シェルの中で実行できます。

```
dbisql -c "uid=dba;pwd=sql;charset=cp1252" -codepage
850 reload.sql
```

データベースに適切な接続パラメータと、*reload.SQL* ファイルの作成に使用したコード・ページ (上記の例では 850) を指定してください。

手順 1 でテーブル・データだけをエクスポートした場合は、Interactive SQL の INPUT 文を使用してそのデータをインポートできます。

INPUT 文の詳細については、『ASA SQL リファレンス・マニュアル』> 「INPUT 文 [Interactive SQL]」を参照してください。

コード・ページと文字セット参照

以下の項では、Adaptive Server Anywhere でサポートされるコード・ページと文字セットについて説明します。

サポートされているコード・ページ

次の表に、Interactive SQL でサポートされているコード・ページを示します。

詳細については、「dbisql コマンド・ライン・ユーティリティを使用して Interactive SQL を開く」700 ページまたは「Interactive SQL ユーティリティのオプション」702 ページを参照してください。

コード・ページ	説明
1252	Windows ラテン文字 1
037	米国、カナダ (2 か国語、フランス語)、オランダ、ポルトガル、ブラジル、オーストラリア
273	IBM オーストリア、ドイツ
277	IBM デンマーク、ノルウェー
278	IBM フィンランド、スウェーデン
280	IBM イタリア
284	IBM カタロニア語／スペイン、スペイン語圏ラテン・アメリカ
285	IBM 英国、アイルランド
297	IBM フランス
420	IBM アラビア語
424	IBM ヘブライ語
437	MS-DOS 米国、オーストラリア、ニュージーランド、南アフリカ
500	EBCDIC 500V1

コード・ページ	説明
737	PC ギリシア文字
775	PC バルト諸語
838	IBM タイ拡張 SBCS
850	MS-DOS ラテン文字 1
852	MS-DOS ラテン文字 2
855	IBM キリル文字
856	IBM ヘブライ語
857	IBM トルコ語
858	Cp850 の変形でユーロ文字を含む
860	MS-DOS ポルトガル語
861	MS-DOS アイスランド語
862	PC ヘブライ語
863	MS-DOS カナダ系フランス語
864	PC アラビア語
865	MS-DOS 北欧
866	MS-DOS ロシア語
868	MS-DOS パキスタン
869	IBM 近代ギリシア語
870	IBM 多言語ラテン文字 2
871	IBM アイスランド
874	IBM タイ
875	IBM ギリシア語
918	IBM パキスタン (ウルドゥ語)
921	IBM ラトビア、リトアニア (AIX、DOS)

コード・ページ	説明
922	IBM エストニア (AIX、DOS)
930	4370 UDC を含む日本語カタカナ漢字、5026 のスーパーセット
933	1880 UDC を含む韓国語、5029 のスーパーセット
935	1880 UDC を含む簡体文字中国語ホスト、5031 のスーパーセット
937	6204 UDC を含む繁体文字中国語ホスト、5033 のスーパーセット
939	4370 UDC を含む日本語ラテン文字漢字、5035 のスーパーセット
942	IBM OS/2 日本語、Cp932 のスーパーセット
942	Cp942 の C 変形
943	IBM OS/2 日本語、Cp932 とシフト JIS のスーパーセット
943	Cp943 の C 変形
948	OS/2 中国語 (台湾)、938 のスーパーセット
949	PC 韓国語
949	Cp949 の C 変形
950	PC 中国語 (香港、台湾)
964	AIX 中国語 (台湾)
970	AIX 韓国語
1006	IBM AIX パキスタン (ウルドゥ語)
1025	IBM 多言語キリル文字 : ブルガリア、ボスニア、ヘルツェゴビナ、マケドニア (FYR)
1026	IBM ラテン文字 5、トルコ
1046	IBM アラビア文字 ― Windows

コード・ページ	説明
1097	IBM イラン (現代ペルシア語) ／ペルシア語
1098	IBM イラン (現代ペルシア語) ／ペルシア語 (PC)
1112	IBM ラトビア、リトアニア
1122	IBM エストニア
1123	IBM ウクライナ
1124	IBM AIX ウクライナ
1140	Cp037 の変形でユーロ文字を含む
1141	Cp273 の変形でユーロ文字を含む
1142	Cp277 の変形でユーロ文字を含む
1143	Cp278 の変形でユーロ文字を含む
1144	Cp280 の変形でユーロ文字を含む
1145	Cp284 の変形でユーロ文字を含む
1146	Cp285 の変形でユーロ文字を含む
1147	Cp297 の変形でユーロ文字を含む
1148	Cp500 の変形でユーロ文字を含む
1149	Cp871 の変形でユーロ文字を含む
1250	Windows 東欧
1251	Windows キリル文字
1253	Windows ギリシア文字
1254	Windows トルコ語
1255	Windows ヘブライ語
1256	Windows アラビア語
1257	Windows バルト諸語
1258	Windows ベトナム語

コード・ページ	説明
1381	IBM OS/2、DOS 中国 (中華人民共和国)
1383	IBM AIX 中国 (中華人民共和国)
33722	IBM-eucJP － 日本語 (5050 のスーパーセット)
ASCII	情報交換用アメリカ標準コード
ISO8859_1	ISO 8859-1 (Latin-1)
UnicodeBig	16 ビット・ユニコード変換フォーマット、ビッグ・エンディアン・バイト順、バイト順マーク付き
UnicodeBigUnmarked	16 ビット・ユニコード変換フォーマット、ビッグ・エンディアン・バイト順
UnicodeLittle	16 ビット・ユニコード変換フォーマット、リトル・エンディアン・バイト順、バイト順マーク付き
UnicodeLittleUnmarked	16 ビット・ユニコード変換フォーマット、リトル・エンディアン・バイト順
UTF8	8 ビット・ユニコード変換フォーマット
UTF-16	16 ビット・ユニコード変換フォーマット、必須の初期バイト順マークで指定されたバイト順
Big5	Big5、中国語 (繁体文字)
Big5_HKSCS	Big5 (香港拡張付き)、中国語 (繁体文字)
Big5_Solaris	Big5 (Solaris zh_TW.BIG5 ロケール用の 7 つの追加 Hanzi 表意文字マッピング付き)
EUC_CN	GB2312、EUC エンコード、中国語 (簡体文字)
EUC_JP	JIS X 0201、0208、0212、EUC エンコード、日本語
EUC_KR	KS C 5601、EUC エンコード、韓国語
EUC_TW	CNS11643 (Plane 1-3)、EUC エンコード、中国語 (繁体字)
GB18030	中国語 (繁体字)、PRC 標準
GBK	GBK、中国語 (繁体字)

コード・ページ	説明
ISCII91	インド語派、ISCII91 エンコード
ISO2022CN	ISO 2022 CN、中国語 (ユニコードへの変換のみ)
ISO2022CN_CNS	ISO 2022 CN 形式の CNS 11643、繁体字中国語 (ユニコードからの変換のみ)
ISO2022CN_GB	ISO 2022 CN 形式の GB 2312、簡体字中国語 (ユニコードからの変換のみ)
ISO2022JP	JIS X 0201、ISO 2022 形式の 0208、日本語
ISO2022KR	ISO 2022 KR、韓国語
ISO8859_2	ISO 8859-2 (Latin-2)
ISO8859_3	ISO 8859-3 (Latin-3)
ISO8859_4	ISO 8859-4 (Latin-4)
ISO8859_5	ISO 8859-5 ラテン／キリル文字アルファベット
ISO8859_6	ISO 8859-6 ラテン／アラビア文字アルファベット
ISO8859_7	ISO 8859-7 ラテン／ギリシャ文字アルファベット
ISO8859_8	ISO 8859-8 ラテン／ヘブライ語アルファベット
ISO8859_9	ISO 8859-9 (Latin-5)
ISO8859_13	ISO 8859-13 (Latin-7)
ISO8859_15_FDIS	ISO 8859-15 (Latin-9)
JIS0201	JIS X 0201、日本語
JIS0208	JIS X 0208、日本語
JIS0212	JIS X 0208、日本語
JISAutoDetect	シフト JIS、EUC-JP、ISO 2022 JP の検出と変換 (ユニコードの変換のみ)
Johab	Johab、韓国語
KOI8_R	KOI8-R、ロシア語

コード・ページ	説明
MS874	Windows タイ語
MS932	Windows 日本語
MS936	Windows 中国語 (簡体字)
MS949	Windows 韓国語
MS950	Windows 中国語 (繁体字)
MacArabic	Macintosh アラビア語
MacCentralEurope	Macintosh ラテン文字 2
MacCroatian	Macintosh クロアチア語
MacCyrillic	Macintosh キリル語
MacDingbat	Macintosh Dingbat
MacGreek	Macintosh ギリシャ語
MacHebrew	Macintosh ヘブライ語
MacIceland	Macintosh アイスランド語
MacRoman	Macintosh Roman
MacRomania	Macintosh ルーマニア語
MacSymbol	Macintosh シンボル
MacThai	Macintosh タイ語
MacTurkish	Macintosh トルコ語
MacUkraine	Macintosh ウクライナ
SJIS	シフト JIS、日本語
TIS620	TIS620、タイ

文字セット・ラベル

次の表に、有効な文字セット・ラベルの値、それに対応する IANA ラベルと説明を示します。

文字セット・ラベル	IANA ラベル	説明
big5	< なし >	中国語 (繁体文字) (CP950 を参照)
cp437	< なし >	IBM CP437 - U.S. コード・セット
cp850	< なし >	IBM CP850 - ヨーロッパ・コード・セット
cp852	< なし >	PC 東ヨーロッパ
cp855	< なし >	IBM PC キリル語
cp856	< なし >	代替ヘブライ語
cp857	< なし >	IBM PC トルコ語
cp860	< なし >	PC ポルトガル語
cp861	< なし >	PC アイスランド語
cp862	< なし >	PC ヘブライ語
cp863	< なし >	IBM PC カナダ系フランス語コード・ページ
cp864	< なし >	PC アラビア語
cp865	< なし >	PC 北ヨーロッパ・ゲルマン系言語
cp866	< なし >	PC ロシア語
cp869	< なし >	IBM PC ギリシャ語
cp874	< なし >	Microsoft タイ語 SB コード・ページ
cp932	windows-31j	Microsoft CP932 = Win31J-DBCS

文字セット・ラベル	IANA ラベル	説明
cp936	< なし >	中国語 (簡体文字)
cp949	< なし >	韓国語
cp950	< なし >	PC (MS) 中国語 (繁体文字)
cp1250	< なし >	MS Windows 東ヨーロッパ言語
cp1251	< なし >	MS Windows キリル語
cp1252	< なし >	MS Windows US (ANSI)
cp1253	< なし >	MS Windows ギリシャ語
cp1254	< なし >	MS Windows トルコ語
cp1255	< なし >	MS Windows ヘブライ語
cp1256	< なし >	MS Windows アラビア語
cp1257	< なし >	MS Windows バルト語派
cp1258	< なし >	MS Windows ベトナム語
deckanji	< なし >	DEC UNIX JIS エンコード
euccns	< なし >	EUC CNS エンコード拡張なし中国語 (繁体文字)
eucgb	< なし >	EUC GB エンコード = 中国語 (簡体文字)
eucjis	euc-jp	Sun EUC JIS エンコード
eucksc	< なし >	EUC KSC 韓国語エンコード (CP949 を参照)
greek8	< なし >	HP ギリシャ語 8
iso_1	iso_8859-1:1987	ISO 8859-1 ラテン語 1
iso15	< なし >	ISO 8859-15 ラテン語 1 ヨーロッパ言語とその他の言語を含む

文字セット・ラベル	IANA ラベル	説明
iso88592	iso_8859-2:1987	ISO 8859-2 ラテン語 2 東ヨーロッパ言語
iso88595	iso_8859-5:1988	ISO 8859-5 ラテン／キリル語
iso88596	iso_8859-6:1987	ISO 8859-6 ラテン／アラビア語
iso88597	iso_8859-7:1987	ISO 8859-7 ラテン／ギリシャ語
iso88598	iso_8859-8:1988	ISO 8859-8 ラテン／ヘブライ語
iso88599	iso_8859-9:1989	ISO 8859-9 ラテン 5 トルコ語
koi8	< なし >	KOI-8 キリル語
mac	macintosh	標準 MAC コード
mac_cyr	< なし >	Macintosh キリル語
mac_ee	< なし >	Macintosh 東ヨーロッパ言語
macgrk2	< なし >	Macintosh ギリシャ語
macturk	< なし >	Macintosh トルコ語
roman8	hp-rpman8	HP Roman-8
sjis	shift_jis	シフト JIS (拡張なし)
tis620	< なし >	TIS-620 標準タイ語
turkish8	< なし >	HP トルコ語 8
utf8	utf-8	文字セットとして扱われる UTF-8

第 12 章

バックアップとデータ・リカバリ

この章の内容

この章では、オペレーティング・システムのクラッシュ、ファイルの破損、ディスク障害、マシン全体の障害からデータを保護する方法について説明します。

この章では、データベースのバックアップを作成する方法、バックアップしたデータをリストアする方法、検証、パフォーマンス、データ保護を考慮してサーバを実行する方法について説明します。

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。

バックアップからデータベースをリストアすることは、データベースに対する「**リカバリ**」の 1 つの側面です。もう 1 つの側面としては、オペレーティング・システムまたはデータベース・サーバのクラッシュ、不適切な停止からのリカバリがあります。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」484 ページ
リカバリとは?	「バックアップとリカバリの概要」484 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」491 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」487 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」487 ページ
バックアップに使用できるツール	「バックアップの作成方法」488 ページ

質問 …	参照先 …
使用可能なバックアップの種類	「バックアップの種類」494 ページ
適切なバックアップの種類	「バックアップ・プロシージャの設計」494 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	「メディア障害からのデータの保護」492 ページ
バックアップの実行方法	「バックアップの概要」490 ページ
バックアップの実行頻度	「バックアップのスケジュール」496 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」496 ページ
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」500 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」502 ページ
テープにバックアップする方法	「データベースを直接テープにバックアップする」537 ページ
バックアップ・スケジュールを計画する方法	「バックアップとリカバリのプランの設計」503 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」395 ページ

質問 …	参照先 …
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	「データベースの妥当性の確認」504 ページ 「データベースの検証」525 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	「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」519 ページ 「トランザクション・ログの検証」527 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	「データ保護を目的としたデータベースの構成」507 ページ
高可用性とマシンの冗長性を確実にする方法	「マシン全体におよぶ障害からの保護」509 ページ 「ライブ・バックアップの作成」539 ページ
バックアップの実行方法	「フル・バックアップの実行」523 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	「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」540 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」541 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」542 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。Adaptive Server 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。この障害は、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。

詳細については、「バックアップの概要」490 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。これには、適切な手順でコンピュータを停止または再起動しなかった場合、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場合、停電の場合などがあります。

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。

リカバリ・オプションの詳細については、「バックアップとリカバリの内部」512 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。詳細については、「コミットされていない操作のリカバリ」546 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 [データベースのバックアップ] ウィザードを使用して、バックアップを作成できます。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] をクリックします。

  詳細については、「データベースを直接テープにバックアップする」537 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 コマンド・ライン・ユーティリティを使用してバックアップを作成できます。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・マシン上のディレクトリ *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ことで、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・マシン上のディレクトリ *c:¥backup* に作成できます。

```
dbbackup -c "connection-string" -s c:¥backup
```

詳細については、「バックアップ・ユーティリティ」646 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・マシンのディレクトリ *c:¥backup* に格納されます。

  ```
  BACKUP DATABASE
   DIRECTORY 'c:¥¥backup'
  ```

  詳細については、『ASA SQL リファレンス・マニュアル』>「BACKUP 文」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。

  詳細については、「バックアップの種類」494 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、DBA 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

データベースが停止しているとき、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完全かつ現行のコピーが保持されています。しかし、データベースの稼動中は、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。

チェックポイントの詳細については、「チェックポイントとチェックポイント・ログ」514 ページと「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」518 ページを参照してください。

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 または Replication Agent 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。

メディア障害から保護するためにトランザクション・ログを使用する方法の詳細については、「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」507 ページ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共に、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。詳細については、「COOPERATIVE_COMMITS オプション [データベース]」 835 ページ と 「DELAYED_COMMITS オプション [データベース]」 843 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。

詳細については、「トランザクション・ログのメディア障害からの保護」 508 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。

データ保護メカニズムの概要については、「障害からのデータの保護」 487 ページを参照してください。

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。

データベース・システムを設定する方法の詳細については、「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」507 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。

詳細については、「トランザクション・ログのメディア障害からの保護」508 ページを参照してください。

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、*dbmisync.exe* 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。

バックアップの概念の詳細については、「バックアップとリカバリの概要」484 ページと「バックアップの概要」490 ページを参照してください。

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。「**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。

バックアップ作成の詳細については、「フル・バックアップの実行」523 ページと「インクリメンタル・バックアップの実行」524 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**　バックアップ・ユーティリティを使用して、クライアント・マシンからオンライン・バックアップを実行できます。サーバ側でバックアップを実行するには、BACKUP 文を実行します。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。

  サーバ側とクライアント側のバックアップの手順は、それぞれ異なります。

- **アーカイブのバックアップとイメージのバックアップ**　「**アーカイブのバックアップ**」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。「**イメージのバックアップ**」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。アーカイブの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、アーカイブのバックアップを使用してください。それ以外の場合は、イメージの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。

  アーカイブのバックアップは、Windows NT/2000/XP 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、イメージのバックアップのみが許可されています。

  アーカイブのバックアップの詳細については、「データベースを直接テープにバックアップする」537 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** マシン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・バックアップをどのようなときに使用するかについては、「マシン全体におよぶ障害からの保護」509 ページを参照してください。

  ライブ・バックアップの作成方法については、「ライブ・バックアップの作成」539 ページを参照してください。

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。アーカイブのバックアップは、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。

詳細については、「データ保護を目的としたデータベースの構成」507 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、バックアップ・コピーの一部をオフサイトに保管してください。

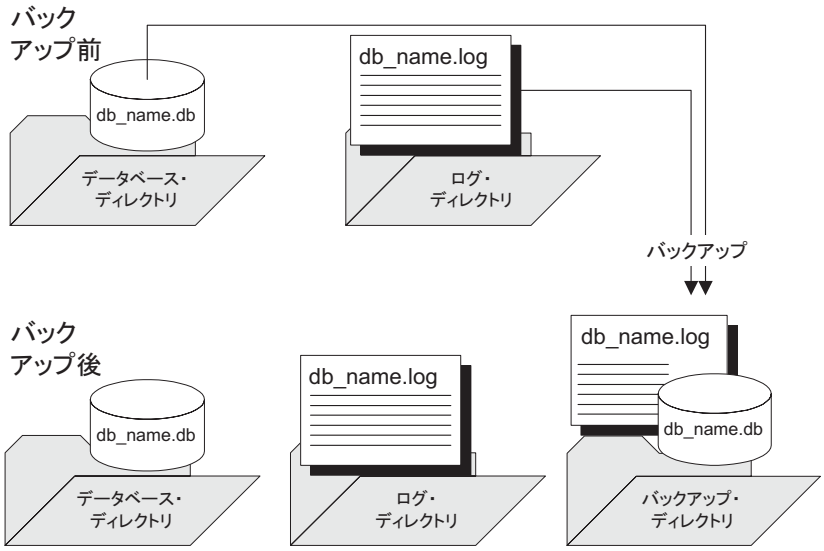
Adaptive Server Anywhere のイベント・スケジュール機能を使用して、スケジュールした時刻に自動的にオンライン・バックアップを実行できます。

バックアップなどのスケジュール操作については、「スケジュールとイベントの使用によるタスクの自動化」395 ページを参照してください。

ディスク領域が十分な場合のバックアップ・スキーマ

運用マシン (データベース・サーバが稼働しているマシン) 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、データベース・ファイルとトランザクション・ログをコピーするだけの簡単なバックアップができ、トランザクション・ログを所定の場所に格納したままにできます。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。

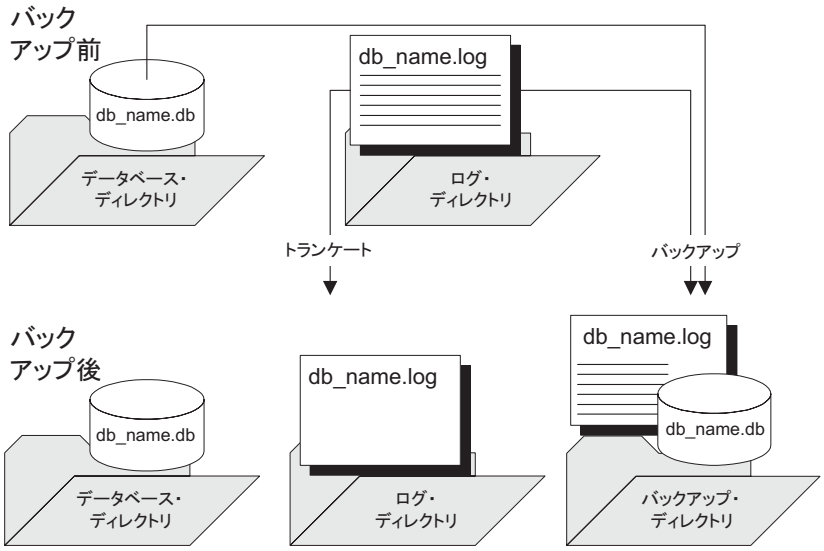


この種のバックアップの実行方法については、「バックアップを作成し、元のトランザクション・ログを継続して使用する」528 ページを参照してください。

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、「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」500 ページで説明しているような適切なスキームを使用してください。

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・ログを削除する」530 ページ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

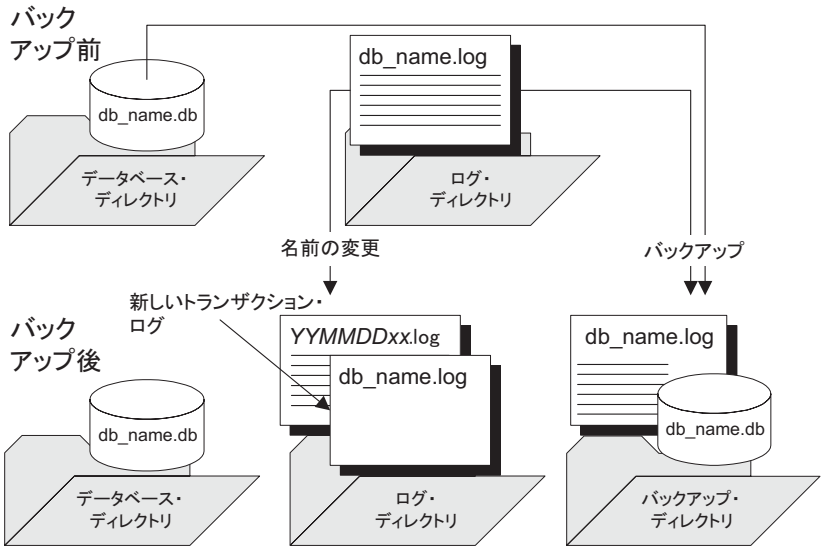
データベースが SQL Remote インストール環境の一部である場合、Message Agent は古いトランザクションにアクセスする必要があります。統合データベースの場合、データベース内に SQL Remote インストール環境全体のマスタ・コピーが格納されるので、完全なバックアップ手順が必要です。

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.exe* 実行プログラム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」532 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・ログの名前を *YYMMDDxx.log* という形式のファイル名に変更します。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 形式のファイル名で年を表すのに 2 桁の数字が使用されていますが、これは 2000 年問題の影響を受けません。この名前は、順番付けのためではなく、識別だけを目的として使用されるからです。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・オプションをオンに設定すると、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の詳細については、「ログ変換ユーティリティ」721 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 オプションをオンに設定すると、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・ログの名前を変更する

-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。

データベースをテープ・ドライブにバックアップする

前述のバックアップの種類はすべて、イメージのバックアップです。各ファイルのバックアップ・コピーもファイルとして格納されます。イメージの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。

アーカイブの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。アーカイブのバックアップは、常にフル・バックアップです。アーカイブのバックアップでは、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。

BACKUP 文を使用すると、アーカイブのバックアップを作成できます。詳細については、「データベースを直接テープにバックアップする」537 ページと 『ASA SQL リファレンス・マニュアル』> 「BACKUP 文」を参照してください。

RESTORE 文を使用すると、バックアップをリストアできます。詳細については、「アーカイブのバックアップのリストア」544 ページと『ASA SQL リファレンス・マニュアル』> 「RESTORE DATABASE 文」を参照してください。

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、Adaptive Server 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。

バックアップの自動化の詳細については、「スケジュールとイベントの使用によるタスクの自動化」395 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。

メディア障害の詳細については、「メディア障害からのデータの保護」492 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のプランの実行については、「バックアップとリカバリ処理の設定」522 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、アプリケーション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。これは、データベースを「**検証**」することで実行できます。この作業には、DBA 権限が必要です。

データベースの検証では、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。データベースの検証では、-f オプションを使用してフル検証を実行しないかぎり、データ、連続したローの参照、外部キー関係は検証されません。

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、そのページが変更されたかディスク上で破壊されています。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データベース・サーバ・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、『ASA SQL リファレンス・マニュアル』> 「VALIDATE CHECKSUM 文」または「検証ユーティリティ」783 ページを参照してください。

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。バックアップしている間に実行中のトランザクションがなかった場合、読み込み専用モードを使用してバックアップ・データベースの有効性をチェックできます。ただし、実行中のトランザクションがあった場合は、データベースの開始時に実行されるデータベース・サーバによるリカバリに任せる必要があります。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。

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。

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作成されません。

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。

詳細については、「データベースの検証」525 ページと「トランザクション・ログの検証」527 ページを参照してください。

読み込み専用データベースの詳細については、「-r サーバ・オプション」206 ページを参照してください。

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 つの物理ディスク・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。

データベースの作成については、『ASA SQL ユーザーズ・ガイド』>「データベースの作成」を参照してください。

トランザクション・ログの格納場所を変更する方法については、「トランザクション・ログの場所の変更」550 ページを参照してください。

トランザクション・ログのメディア障害からの保護

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 オプションを指定すると、ファイル・システム・フルの状態が発生したときにコールバック関数を実装できます。

詳細については、「-fc サーバ・オプション」186 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 NT/2000/XP 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。

関連項目

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。

詳細については、「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」509 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」687 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」752 ページを参照してください。

マシン全体におよぶ障害からの保護

「**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・サーバを実行するマシン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・マシン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・バックアップの作成については、「ライブ・バックアップの作成」539 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方法については、「ライブ・バックアップのリカバリ」543 ページを参照してください。

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・コピーを作成するように見えます。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マシンで実行される** トランザクション・ログ・ミラーの場合、別のマシン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、マシン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。

  別のマシンでバックアップ・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 Adaptive Server 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。

- **ライブ・バックアップは、マシン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、マシン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 台のマシン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方法については、「ライブ・バックアップの作成」539 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。バックアップの実行中、他の接続によって変更されるページは変更前にデータベース・ファイルでなくテンポラリ・ファイル内に保存されるので、バックアップ・イメージはチェックポイント時点のものとして作成されます。

2. 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 K 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. バージョン 8.0 以降の Adaptive Server Anywhere で作成されたデータベースでは、バックアップ命令によりトランザクション・ログのトランケートや名前の変更が要求される場合、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・ログに送られます。

   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では、バックアップ命令によりトランザクション・ログのトランケートや名前の変更が要求される場合、データベース・サーバはコミットされていないトランザクションがなくなるまで待機してから、ログ・ファイルのトランケートまたは名前の変更を実行します。データベースがビジーの場合は、この待機が非常に長くなることがあります。

   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」494 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 文のほかにデータ定義文も含まれます。

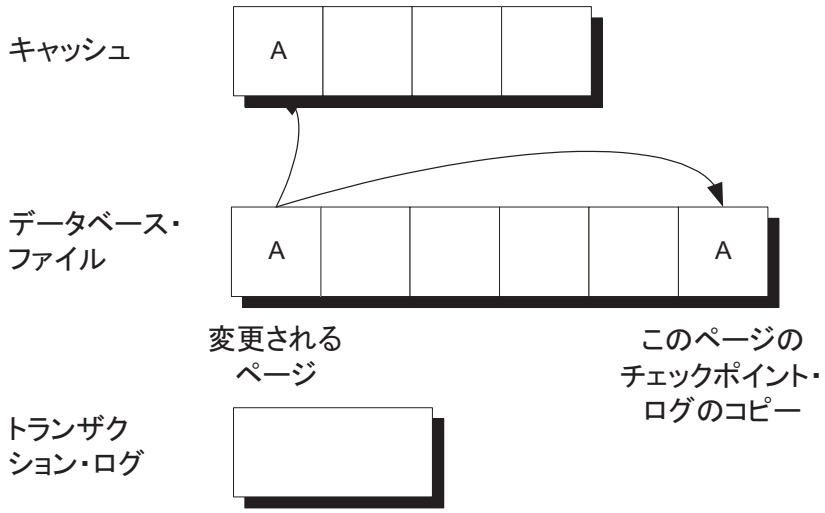
リカバリ (バックアップのリストアを含む) の間、データベースの別のユーザに許可されているアクションはありません。

チェックポイントとチェックポイント・ログ

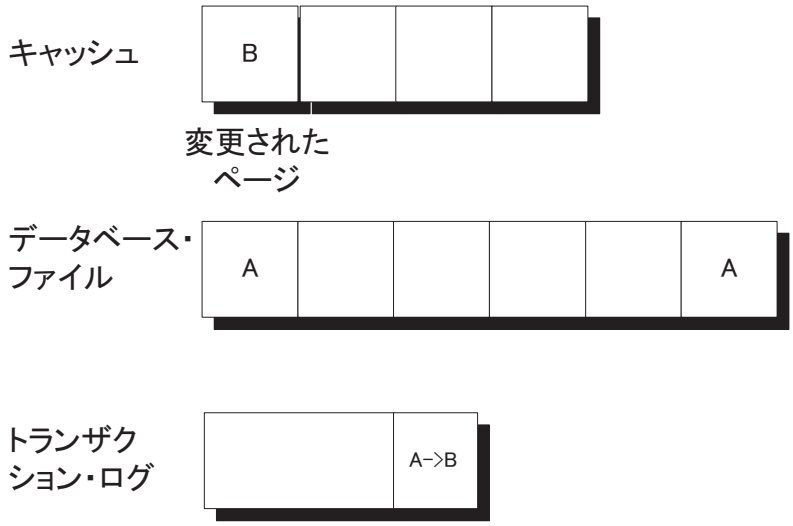
データベース・ファイルはページで構成されています。ページとは、ハード・ディスク内の決められたサイズの領域です。チェックポイント・ログは、データベース・ファイルの最後にあります。セッション中は必要に応じてチェックポイント・ログにページが追加され、セッションの最後にはチェックポイント・ログ全体が削除されます。

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。

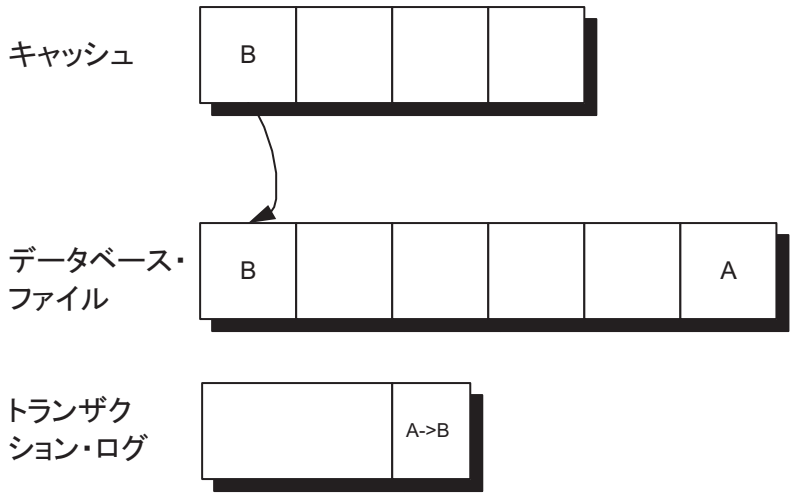
- ページがメモリに読み込まれ、データベース・キャッシュ内に格納されます。
- 元のページのコピーが作成されます。これらのコピーされたページを「**チェックポイント・ログ**」と呼びます。



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。

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」518 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。

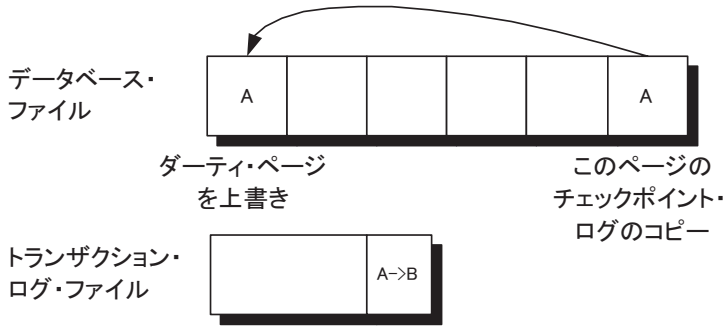
トランザクション処理の詳細については、『ASA SQL ユーザーズ・ガイド』>「トランザクションと独立性レベル」を参照してください。

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。



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。

  詳細については、「-gc サーバ・オプション」190 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。

  詳細については、「-gr サーバ・オプション」196 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ためのデータベース・オプションは 2 つあります。CHECKPOINT_TIME は、チェックポイント間の最大時間を決定します。RECOVERY_TIME は、システム障害時のリカバリまでの最大時間を決定します。

詳細については、「CHECKPOINT_TIME オプション [データベース]」829 ページ と 「RECOVERY_TIME オプション [データベース]」890 ページ を参照してください。

ディスクへのダーティ・ページの書き込みは、「**アイドル I/O タスク**」と呼ばれるサーバ内のタスクによって実行されます。このタスクは、他のデータベース・タスクと処理時間を共有しています。

ダーティ・ページのページ数にはスレッショルドが定められており、ページ数がスレッショルドに満たないうちはデータベース・ページの書き込みは行われません。

データベースがビジーで、緊急度が低く、キャッシュにダーティ・ページが少ししかない場合、アイドル I/O タスクの優先度は非常に低く設定され、ダーティ・ページの書き込みは行われません。

緊急度が 30% を超えると、アイドル I/O タスクの優先度が上がります。間隔を置いて、優先度が再び上がります。緊急度が高くなるほど、サーバは処理の重点をダーティ・ページの書き込みに移行します。これは、ページ数がスレッショルドよりも小さくなるまで続きます。ただし、ダーティ・ページのページ数がスレッショルドよりも多い場合のみ、サーバはアイドル I/O タスク中にページの書き出しを行います。

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。

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、WITH DATA CHECK 句を使用する従来の検証と同じように、テーブルの各ローが読み込まれて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。

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。

エクスプレス・チェック・オプションの詳細については『ASA SQL リファレンス・マニュアル』>「VALIDATE TABLE 文」、「dbvalid コマンド・ライン・ユーティリティを使用したデータベースの検証」785 ページ、および『ASA SQL リファレンス・マニュアル』>「sa_validate システム・プロシージャ」を参照してください。

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。

詳細については、「バックアップとリカバリのプランの設計」503 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。

バックアップの実行の詳細については、「フル・バックアップの実行」523 ページと「インクリメンタル・バックアップの実行」524 ページを参照してください。

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」494 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。

詳細については、「データベースの検証」525 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」528 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」530 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」532 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」504 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」494 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については、次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」528 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」530 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」532 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 *asademo.db* と *asademo.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」536 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。詳細については、「データベースの妥当性の確認」504 ページを参照してください。

バックアップ操作の概要については、「フル・バックアップの実行」523 ページ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. 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. 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。
3. [ファイル] ー [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。
4. 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。

❖ 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)。

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 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、『ASA SQL リファレンス・マニュアル』>「sa_validate システム・プロシージャ」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)。

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 dbvalid ユーティリティを実行します。

```
dbvalid -c "connection_string"
```

詳細については、「検証ユーティリティ」783 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。

読み込み専用モードによるデータベース実行の詳細については、「-r サーバ・オプション」206 ページを参照してください。

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。テーブルの有効性を検証するには、DBA 権限を持っているかテーブルの所有者である必要があります。

❖ テーブルの妥当性を検証するには、次の手順に従います (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。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。

トランザクション・ログの検証

トランザクション・ログ・ファイルの検証処理は、運用データベースで使用されている (オンライン) か使用されていない (オフライン、またはバックアップ・コピー) かによって異なります。

❖ **オンライン・トランザクション・ログを検証するには、次の手順に従います。**

- ミラーされたトランザクション・ログを使ってデータベースを実行します。データベース・サーバは、データベースを起動するたびに自動的にトランザクション・ログを検証します。

❖ **オフライン・トランザクション・ログまたはバックアップされたトランザクション・ログを検証するには、次の手順に従います。**

- ログ・ファイルに対してログ変換ユーティリティ (dbtran) を実行します。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。

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」497 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 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。

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 とは別のマシン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・マシン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。

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・マシン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」498 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 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。

詳細については、「バックアップの内部メカニズム」512 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。

詳細については、「未処理のトランザクションがある接続の判断」534 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」500 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 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。

詳細については、「バックアップの内部メカニズム」512 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。

詳細については、「未処理のトランザクションがある接続の判断」534 ページ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の判断

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、Adaptive Server 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. **UncmtOps** 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。

   詳細については、『ASA SQL リファレンス・マニュアル』>「sa_conn_info システム・プロシージャ」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 (Adaptive Server Anywhere コンソール・ユーティリティの場合)。

1. Adaptive Server 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"
   ```

詳細については、「Adaptive Server Anywhere コンソール・ユーティリティ」644 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、asademo データベース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=asademo" -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* 形式のファイル名で年を表すのに 2 桁の数字が使用されていますが、これは 2000 年問題の影響を受けません。この名前は、順番付けのためではなく、識別だけを目的として使用されるからです。

データベースを直接テープにバックアップする

アーカイブのバックアップでは、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。

詳細については、「バックアップの種類」494 ページを参照してください。

❖ **アーカイブの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 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。

[データベースのバックアップ] 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。

❖ アーカイブの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。

アーカイブのバックアップのリストアについては、「アーカイブのバックアップのリストア」544 ページを参照してください。

例 次の文では、Windows NT マシン上で最初のテープ・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 NT 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

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・バックアップの詳細については、「マシン全体におよぶ障害からの保護」509 ページを参照してください。

❖ 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・マシン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・マシンを設定します。たとえば、Adaptive Server Anywhere がセカンダリ・マシン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・マシンにフル・バックアップを実行します。
3. セカンダリ・マシン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¥filename.log -c "connection_string"
   ```

通常、dbbackup ユーティリティはセカンダリ・マシンから実行してください。

プライマリ・マシンを使用できなくなった場合、セカンダリ・マシン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

リカバリ処理の過程は、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。

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」494 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。

各ログ・ファイルについて、日付順に次の操作を行います。

- ログ・ファイルをリカバリ・ディレクトリにコピーします。

- 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。

  ```
  dbeng9 db_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。

5 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。

```
dbeng9 db_name.db -a db_name.log
```

6 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。

詳細については、「データベースの検証」525 ページを参照してください。

7 リカバリ後のバックアップを作成します。

8 データベース・ファイルを運用ディレクトリに移します。

9 運用データベースへのユーザ・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。

詳細については、「トランザクション・ログのメディア障害からの保護」508 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、または名前を変更します。

3 -f オプションを使って、データベースを再起動します。

```
dbeng9 asademo.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。

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

❖ 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 *asademo.log* を変換して、*asademo.SQL* ファイルに出力します。

```
dbtran asademo.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・マシンとは別のマシンに作成されます。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・マシンに Adaptive Server 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、「マシン全体におよぶ障害からの保護」509 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　セカンダリ・マシン上のデータベース・サーバをトランザクション・ログ適用 (-a) オプションを使って起動し、トランザクション・ログを適用してデータベースを更新します。

```
dbeng9 asademo.db -a filename.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。

2 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、すべて現行のトランザクション・ログに追加されます。

アーカイブのバックアップのリストア

アーカイブのバックアップ (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。

アーカイブの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」537 ページを参照してください。

❖ アーカイブの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 権限を使用してデータベースに接続します。

2 左ウィンドウ枠で Adaptive Server Anywhere プラグインを選択し、右ウィンドウ枠で [ユーティリティ] タブをクリックします。

3 右ウィンドウ枠の [データベースのリストア] をダブルクリックします。

[データベース・リストア] ウィザードが表示されます。

4 ウィザードの指示に従います。

❖ アーカイブの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。

```
dbeng9 -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場所 (デフォルト) にリストアする、または RENAME 句を使用して別のマシンの別のデバイス名を使ってリストアするように選択できます。

詳細については、『ASA SQL リファレンス・マニュアル』> 「RESTORE DATABASE 文」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、ファイル *c:¥backup¥archive.1* のアーカイブのバックアップのデータベースをデータベース・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、『ASA SQL リファレンス・マニュアル』> 「RESTORE DATABASE 文」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。この際、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。

❖ 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 次のいずれかを行います。

- データベースに接続されている場合は、左ウィンドウ枠で Adaptive Server Anywhere プラグインを選択し、右ウィンドウ枠で [ユーティリティ] タブをクリックします。右ウィンドウ枠で、[ログ・ファイルの変換] をダブルクリックします。
- データベースに接続していない場合は、[ツール] ー [Adaptive Server Anywhere 9] ー [ログ・ファイルの変換] の順に選択します。

  [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」721 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

Adaptive Server Anywhere 8.0.0 以降では、トランザクションが複数のログ・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。

dbtran ユーティリティを使用してトランザクション・ログ・ファイルが生成された順番を判別できます。このユーティリティが生成した SQL スクリプトには、トランザクション・ログ内の開始時のログ・オフセットが表示されます。これによって複数のログ・ファイルが生成された順番を効果的に判断でき、確実に正しい順番でファイルを適用できます。このスクリプトを使用すると、トランザクション・ログに複数のトランザクション・ログにまたがるトランザクションが含まれているか否かも判別できます。

詳細については、「トランザクション・ログ・ユーティリティ」752 ページを参照してください。

複数のトランザクション・ログからデータベースをリカバリする場合にデータの整合性を維持するためには、-a サーバ・オプションを使用して各ログに対して個別にデータベースのバックアップ・コピーを適用するか、ログ・トランザクション・ユーティリティ (dbtran) を使用して 1 つまたは複数のトランザクション・ログを *.SQL* ファイルに変換してデータベースのバックアップ・コピーを適用できるようにします。

-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・ログがある場合、ファイルを 1 つずつ正しい順番 (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バックアップ・コピーに適用します。
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。

詳細については、「-a リカバリ・オプション」227 ページを参照してください。

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・サーバを起動して、*backupasademo.log* というバックアップ・トランザクション・ログを、*backupasademo.db* というデータベースのバックアップ・コピーに適用します。

```
dbeng9 -a backupasademo.log backupasademo.db
```

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、*asademo.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。

```
dbeng9 -a asademo.log backupasademo.db
```

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。

詳細については、「トランザクション・ログ・ユーティリティ」752 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 ステップ 1 で生成された *.SQL* ファイルを、Interactive SQL からバックアップ・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. *backupasademo.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。

   ```
   dbeng9 backupasademo.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;eng=backupasademo" 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」507 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 次のいずれかを行います。

- すでにログ・ファイルに関連するデータベースに接続されている場合は、左ウィンドウ枠で Adaptive Server Anywhere プラグインを選択し、右ウィンドウ枠で [ユーティリティ] タブをクリックします。右ウィンドウ枠で、[ログ・ファイル設定の変更] をダブルクリックします。
- データベースに接続していない場合は、[ツール] ー [Adaptive Server Anywhere 9] ー [ログ・ファイル設定の変更] の順に選択します。

  [ログ・ファイル設定の変更] 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。

❖ 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
```

dblog オプションの詳細については、「トランザクション・ログ・ユーティリティのオプション」755 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」508 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 次のいずれかを行います。

- データベースに接続されている場合は、左ウィンドウ枠で Adaptive Server Anywhere プラグインを選択し、右ウィンドウ枠で [ユーティリティ] タブを開きます。右ウィンドウ枠で、[データベースの作成] をダブルクリックします。
- データベースに接続していない場合は、[ツール] ー [Adaptive Server Anywhere 9] ー [データベースの作成] の順に選択します。

  [データベース作成] 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。

❖ 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 句を指定して実行します。

  詳細については、『ASA SQL リファレンス・マニュアル』> 「CREATE DATABASE 文」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
```

初期化オプションの詳細については、「初期化ユーティリティのオプション」690 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」508 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　次のいずれかを行います。

- すでにログ・ミラーに関連するデータベースに接続されている場合は、左ウィンドウ枠で Adaptive Server Anywhere プラグインを選択し、右ウィンドウ枠で [ユーティリティ] タブを開きます。右ウィンドウ枠で、[ログ・ファイル設定の変更] をダブルクリックします。
- データベースに接続していない場合は、[ツール] ― [Adaptive Server Anywhere 9] ― [ログ・ファイル設定の変更] の順に選択します。

  [ログ・ファイル設定の変更] 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。

❖ **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
```

dblog オプションの詳細については、「トランザクション・ログ・ユーティリティのオプション」755 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。

第 3 部　パーミッションとレプリケーション

この項では、ユーザ ID とパーミッションを使用してセキュア・データベースを管理する方法について説明します。また、Replication Server を使用してデータをレプリケートする方法についても説明します。

第 13 章

ユーザ ID とパーミッションの管理

この章の内容

データベースの各ユーザは、データベースに接続するときに入力するユーザ ID と呼ばれる名前が必要です。この章では、ユーザ ID の管理方法について説明します。

データベースのパーミッションの概要

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。

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。

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 ID の同義語として「**ユーザ**」という用語を使用します。パーミッションの付与と取り消しは、ユーザ ID 単位で行います。

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。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。この章では、ユーザのグループについても説明します。

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。

- ユーザによって加えられた変更を、ログ変換ユーティリティがトランザクション・ログからユーザ別に抽出できます。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、個々のユーザ ID に関するより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 オプションがオフの場合) 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、使用可能なユーザ ID が 1 つ作成されます。この最初のユーザ ID は **DBA** であり、パスワードの初期設定は **SQL** です。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。

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。

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。この機能の使用方法については、「パスワードのないグループ」584 ページを参照してください。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人物として扱います。

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

パーミッション	説明
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。テーブルについては、カラムのセットに対してこのパーミッションを与えることもできる。
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。

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右クリックし、ポップアップ・メニューで [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

- 『ASA SQL ユーザーズ・ガイド』> 「データベース・オブジェクトのプロパティの設定」
- 「データベース・オプション」795 ページ

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。

  詳細については、「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」581 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。

  詳細については、「個別のユーザ ID とパーミッションの管理」563 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。

  詳細については、『SQL Remote ユーザーズ・ガイド』> 「SQL Remote パーミッションの管理」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・テーブルの表示
- 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。これについては、「特殊なグループ」585 ページを参照してください。

❖ 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] ー [ユーザ] を選択します。

[ユーザ作成] 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 ID 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

- 『ASA SQL リファレンス・マニュアル』> 「GRANT 文」

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、セミコロンを含めることはできません。

128 以降の 8 ビット ASCII 文字でパスワードを構成した場合、文字セット変換がオンで文字セットが一致しない場合に正確に動作しないことがあるので、パスワードは 7 ビット ASCII 文字で構成することをおすすめします。

128 以降の文字でパスワードを構成する場合、次のようにします。

- 接続文字列に **CharSet=none** と指定して、文字セット変換をオフにします。
- CharSet 接続パラメータを使用して、文字セットを指定します。

  指定できる文字セットの詳細については、「文字セット・ラベル」480 ページを参照してください。
- -ct- オプションを指定して、データベース・サーバでの文字セット変換をオフにします。

  詳細については、「-ct サーバ・オプション」174 ページを参照してください。

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、次に示す文は、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、ユーザ ID **DBA** のパスワードを **new_password** に変更します。

```
GRANT CONNECT TO DBA
 IDENTIFIED BY new_password
```

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 権限のあるユーザ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 userid
```

DBA 権限の場合は、次の SQL 文を使います。

```
GRANT DBA TO userid
```

注意

- DBA または RESOURCE の権限をデータベース・ユーザに与えられるのは 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 権限を持つユーザには、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。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することができます。
- 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与えます。

❖ 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. テーブルを右クリックし、ポップアップ・メニューで [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 [References]、[Select]、[Update] 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 権限を使用するか、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。

詳細については、『ASA SQL リファレンス・マニュアル』> 「GRANT 文」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 権限を持つユーザか 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
```

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 権限を持つユーザか 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 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、SELECT と 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

- 『ASA SQL リファレンス・マニュアル』> 「GRANT 文」

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」567 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に対してのみ与えられます。テーブルの場合と異なり、パーミッションをカラムごとに与えることはできません。

❖ 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　ビューを右クリックし、ポップアップ・メニューから [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

- 『ASA SQL リファレンス・マニュアル』> 「GRANT 文」

パーミッションを付与する権利をユーザに付与する

「テーブルに対するパーミッションの付与」567 ページで説明した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。グループに対するこの機能については、「グループのパーミッション」583 ページを参照してください。

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。

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 権限を持つユーザか 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
```

詳細については、『ASA SQL リファレンス・マニュアル』> 「GRANT 文」を参照してください。

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 (プロシージャを作成したユーザ ID) 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、「テーブルに対するパーミッションの付与」567 ページで説明した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　プロシージャを右クリックし、ポップアップ・メニューで [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、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。
- ユーザのとなりの [Execute] 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] 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 権限を使用するか、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 権限を持つユーザか my_procedure 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
```

プロシージャの実行パーミッション

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実行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れば、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

- 『ASA SQL リファレンス・マニュアル』> 「GRANT 文」

トリガの実行パーミッション

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。

トリガのパーミッションの詳細については、『ASA SQL ユーザーズ・ガイド』> 「トリガを実行するためのパーミッション」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 つは定義しない限り、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. 対象のユーザを右クリックし、ポップアップ・メニューで [リモート・ユーザに変更] を選択します。

   [ユーザをリモート・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 リモート・ユーザ] フォルダを開きます。
2. 対象のリモート・ユーザを右クリックし、ポップアップ・メニューで [リモートの取り消し] を選択します。

詳細については、『SQL Remote ユーザーズ・ガイド』> 「SQL Remote の概念」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。

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」567 ページ
- 「ビューに対するパーミッションの付与」569 ページ
- 「プロシージャに対するパーミッションの付与」572 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. 対象のユーザを右クリックし、ポップアップ・メニューで [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

- 『ASA SQL リファレンス・マニュアル』> 「REVOKE 文」
- 「ユーザ・パーミッションの取り消し」576 ページ
- 「データベースからグループを削除する」586 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. 対象のユーザを右クリックし、ポップアップ・メニューで [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. 対象のユーザを右クリックし、ポップアップ・メニューから [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。個別のユーザと同様に、ユーザ ID でグループを識別します。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。

DBA、RESOURCE、および GROUP パーミッション

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。DBA、RESOURCE、GROUP の各パーミッションについては継承されません。これらのパーミッションは、必要に応じて各ユーザ ID に個々に割り当てる必要があります。

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、「個別のユーザ ID とパーミッションの管理」563 ページで説明したコマンドを使用して通常のユーザとまったく同じ方法で実行します。

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」584 ページを参照してください。

データベース・オブジェクト・プロパティの変更については、『ASA SQL ユーザーズ・ガイド』>「データベース・オブジェクトのプロパティの設定」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」575 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] ー [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。まず、この文で使用するユーザ ID を作成しておく必要があります。

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

- 『ASA SQL リファレンス・マニュアル』> 「GRANT 文」
- 「新しいユーザの作成」563 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プロパティ・シートでグループ・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することができます。

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・メニューで [新規] ー [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

- 『ASA SQL リファレンス・マニュアル』> 「GRANT 文」
- 「新しいユーザの作成」563 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。

❖ 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. 対象のグループを右クリックし、ポップアップ・メニューで [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右クリックして、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

- 『ASA SQL リファレンス・マニュアル』> 「REVOKE 文」
- 「新しいユーザの作成」563 ページ
- 「データベースからユーザを削除する」577 ページ
- 「データベースからグループを削除する」586 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。

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えます。次のクエリを例にとります。

```
SELECT * FROM SYSGROUPS
```

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、このクエリは常にビュー SYS.SYSGROUPS を検索します (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。

例については、「データベース・オブジェクトの名前とプレフィクス」588 ページの項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。このユーザ ID は、グループのメンバシップを与えたり取り消したりできます。またこのユーザは、グループのユーザ ID の名のもとに作成されたデータベース内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することができます。

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。それには、次のように GRANT CONNECT 文をパスワードなしで入力します。

```
GRANT CONNECT
 TO personnel
```

これにより、ユーザ ID personnel が作成されます。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、『ASA SQL リファレンス・マニュアル』> 「システム・テーブル」と『ASA SQL リファレンス・マニュアル』> 「システム・ビュー」の章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 対象のグループを右クリックし、ポップアップ・メニューで [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

- 『ASA SQL リファレンス・マニュアル』> 「REVOKE 文」
- 「ユーザ・パーミッションの取り消し」576 ページ
- 「データベースからユーザを削除する」577 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、『ASA SQL リファレンス・マニュアル』>「識別子」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 employee
```

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。サンプル・データベースのテーブルの所有者は、ユーザ ID DBA です。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.employee
```

この employee 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。

ビューの作成方法については、『ASA SQL ユーザーズ・ガイド』>「ビューの編集」を参照してください。

パーミッションの詳細については、「ビューに対するパーミッションの付与」569 ページを参照してください。

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が、他部署の従業員のデータにアクセスする理由はありません。

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 権限を持つユーザ ID としてログインしている間に、次を入力します。

```
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 emp_sales AS
   SELECT emp_id, emp_fname, emp_lname
   FROM DBA.employee
   WHERE dept_id = 200
```

   ユーザ ID SalesManager がビューを使えるように、テーブルは所有権を明確にして DBA.employee とする必要があります。そうしないと、SalesManager がビューを使うとき、SELECT 文はユーザ ID が認識しないテーブルを参照してしまいます。

3. SalesManager にビューを見るパーミッションを与えます。

```
GRANT SELECT
 ON emp_sales
 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 order_summary AS
   SELECT order_date, region, sales_rep,
company_name
   FROM DBA.sales_order
     KEY JOIN DBA.customer
```

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 order_summary
 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.emp_sales;
 SELECT *
 FROM DBA.order_summary;
```

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 DBA.employee;
 SELECT * FROM DBA.sales_order
```

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、『ASA SQL ユーザーズ・ガイド』> 「ビューの使い方」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。「プロシージャに対するパーミッションの付与」572 ページで説明したように、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。

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。

Adaptive Server 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、その作業は行えます。

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、その作業を行えます。

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 を大量に使用するのを防ぐことができます。

Adaptive Server 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*]

オプションの詳細については、「データベース・オプション」795 ページを参照してください。

SET OPTION 文については、『ASA SQL リファレンス・マニュアル』>「SET OPTION 文」を参照してください。

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。

- **JAVA_HEAP_SIZE**　接続単位で Java アプリケーションに割り付けるメモリの最大サイズ (バイト単位) を設定します。
- **MAX_CURSOR_COUNT**　接続用のカーソルの数を制限します。
- **MAX_STATEMENT_COUNT**　接続用の準備文の数を制限します。
- **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。

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。

システム・テーブルのユーザとパーミッション

データベースのシステム・テーブルとシステム・ビューは、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を格納します。

これらのテーブルの詳細については、『ASA SQL リファレンス・マニュアル』>「システム・テーブル」を参照してください。

特殊なユーザ ID SYS は、システム・テーブルを所有します。ユーザ ID SYS を使用して接続することはできません。

DBA は、データベース内の他のテーブルに対してアクセス権を持っているように、すべてのシステム・テーブルに対しても SELECT アクセス権を持っています。他のユーザのシステム・テーブルへのアクセス権は制限されています。たとえば、SYS.SYSUSERPERM テーブルには、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・テーブルの概要を示します。ユーザ ID SYS はリストされたすべてのテーブルとビューを所有し、その修飾された名前は SYS.SYSUSERPERM などになります。

これらのテーブルに対して適切な SELECT クエリを使えば、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。

テーブル	デフォルト	目次
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
SYSGROUP	PUBLIC	各グループのメンバごとに 1 ロー

テーブル	デフォルト	目次
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
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
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
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

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。

ビュー	デフォルト	目次
SYSUSERAUTH	DBA のみ	ユーザ ID 以外のすべての SYSUSERPERM の情報
SYSUSERPERMS	PUBLIC	パスワード以外のすべての SYSUSERPERM の情報
SYSUSERLIST	PUBLIC	パスワード以外のすべての SYSUSERAUTH の情報
SYSGROUPS	PUBLIC	SYSGROUP の情報を読みやすくしたもの
SYSTABAUTH	PUBLIC	SYSTABLEPERM の情報を読みやすくしたもの
SYSCOLAUTH	PUBLIC	SYSCOLPERM の情報を読みやすくしたもの
SYSPROCAUTH	PUBLIC	SYSPROCPERM の情報を読みやすくしたもの

これらのテーブルとビュー以外に、データベースの各オブジェクトの情報を含むテーブルとビューがあります。

第 14 章

Replication Server を使用したデータのレプリケート

この章の内容

この章では、Replication Server を使用して Adaptive Server Anywhere データベースと他のデータベースの間でデータをレプリケートする方法について説明します。レプリケーション・システム内の他のデータベースは、Adaptive Server Anywhere データベースでも他の種類のデータベースでもかまいません。

始める前に

この章は、Adaptive Server 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。

レプリケーションの概要

データの「レプリケーション」とは、物理的にまったく異なるデータベース間でデータを共有することです。共有データにどのデータベースで変更を加えても、それはレプリケーション・システムにある他のデータベースに正確にコピーされます。

データのレプリケーションは、データベース・ユーザに大きな利益をもたらします。

データ可用性

単一の統合データベースでは高コストで信頼性が低く接続速度が遅い可能性がありますが、レプリケーションはローカルでデータが使用できるようにします。データにローカルでアクセスできると、長距離ネットワークの接続障害が発生したような場合でも、いつでもデータを利用することが可能です。

応答時間

レプリケーションは、次の 2 つの理由で、データ要求に対する応答時間を短縮しています。まず、WAN にアクセスせずにローカル・サーバで要求の処理を行うので、検索速度が上がります。次に、ローカル処理によって統合データベース・サーバの作業が軽減されるので、プロセッサ競合時間が減ります。

Sybase のレプリケーション・テクノロジ

Sybase は、Adaptive Server Anywhere のために次のレプリケーション・テクノロジを用意しています。

- **Mobile Link** Mobile Link は、統合データベース・サーバと多数のリモート・データベース (通常は、モバイル・データベースが多数含まれています) との間の双方向レプリケーション用に設計されています。
- **SQL Remote** Mobile Link と同じように、SQL Remote も、統合データベースと多数のリモート・データベース (通常は、モバイル・データベースが多数含まれています) との間の双方向レプリケーション用に設計されています。リモート・サイトで必要な管理とリソースは最小限で済みます。

- **Replication Server**　Replication Server は、比較的少数のデータ・サーバ間で行われるレプリケーション用に設計されています。このテクノロジでは、プライマリ・データとレプリケート・データの間のタイムラグは一般的に数秒であり、通常は各サイトに管理者がいます。

どちらのレプリケーション・テクノロジにも固有のマニュアルがあります。この章では、Replication Server によって Adaptive Server Anywhere を使用する方法について説明します。

SQL Remote については、『SQL Remote ユーザーズ・ガイド』を参照してください。

Mobile Link の詳細については、『Mobile Link 同期ユーザーズ・ガイド』を参照してください。

レプリケート・サイトとプライマリ・サイト

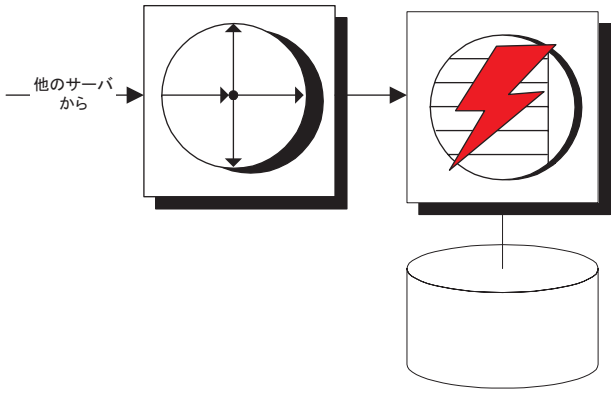
Replication Server のインストールでは、データベース間で共有されるデータは「**レプリケーション・サブスクリプション**」に配列されます。

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

Adaptive Server Anywhere は、レプリケート・サイトとして使用できます。追加のコンポーネントは必要ありません。

次の図は、Adaptive Server 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 Adaptive Server Anywhere に接続して、変更を適用します。
- Adaptive Server Anywhere がデータベースに変更を加えます。

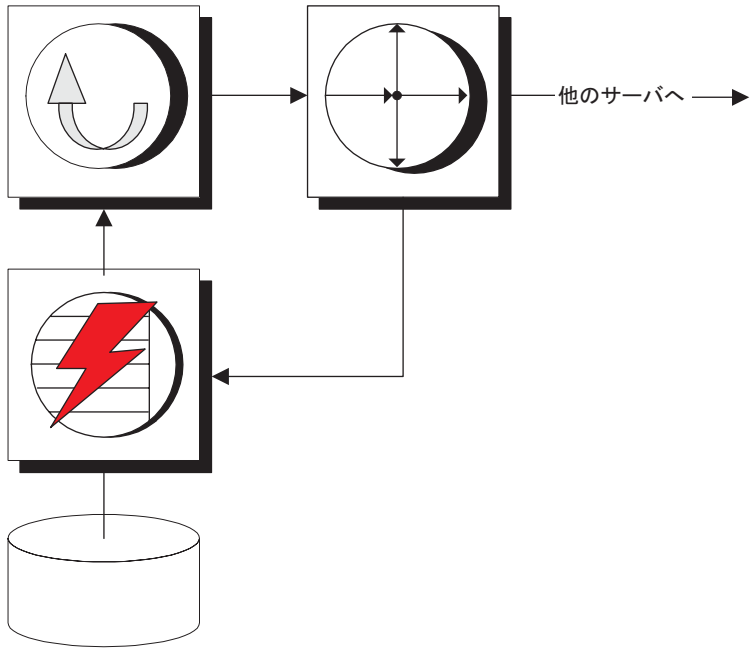
非同期プロシージャ・コール (APC)

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

Adaptive Server 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、Log Transfer Manager (LTM) または Replication Agent 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、Adaptive Server 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。



- Adaptive Server Anywhere データベース・サーバはデータベースを管理します。
- Adaptive Server 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 Adaptive Server 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」627 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：Adaptive Server Anywhere データベースのセット・アップ

この項では、レプリケーション用に Adaptive Server 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。

- 作成しておいたチュートリアル・ディレクトリ (*c:¥tutorial* 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。

- 作成しておいたチュートリアル・ディレクトリ (*c:¥tutorial* など) から次のコマンドを入力します。

  ```
  dbinit repdb
  ```

  これによって、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上で実行されるデータベース・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。

```
dbsrv9 -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。

```
dbsrv9 -x tcpip(port=2639) -n REPSV repdb.db
```

次の作業 次は、各 Adaptive Server 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。NetWare ユーザと UNIX ユーザの場合は、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」140 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、Adaptive Server 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。Adaptive Server Anywhere に付属している SQL コマンド・ファイル *rssetup.sql* を使用して、これらの作業を実行します。

rssetup.sql コマンド・ファイルは、Adaptive Server 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、ユーザ ID **DBA** とパスワード **SQL** を使用して Adaptive Server 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 "path¥rssetup.sql"
   ```

   path には Adaptive Server Anywhere インストール・ディレクトリが入ります。

   または、[ファイル] ー [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 が実行するアクション

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、**dbmaint** という名前のユーザを作成します。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、**sa** という名前のユーザを作成します。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、Adaptive Server 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。Adaptive Server Anywhere では、現在のところ、NULL パスワードを承認しません。

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 のユーザ ID 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

Adaptive Server Anywhere では、すべての識別子に大文字と小文字の区別がありません。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、Adaptive Server 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。

Adaptive Server Anywhere では、明示的にパスワードの大文字と小文字を区別するかどうかを設定しない限り、データベースがこれを決定します。大文字と小文字を区別するデータベースで、大文字と小文字を区別しないパスワードを使用することができ、またその逆も可能です。パスワードで使用される拡張文字は、パスワードの大文字小文字の区別設定にかかわらず、大文字と小文字を区別します。ユーザ ID と識別子については、すべての Adaptive Server 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、『ASA SQL リファレンス・マニュアル』>「CREATE DATABASE 文」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、ALTER TABLE 文を使用するテーブルでは、REPLICATE フラグを ON に設定してください。

```
ALTER TABLE news
 REPLICATE ON
 go
```

これは、Adaptive Server Enterprise のテーブル上で **sp_setreplicate** または **sp_setreptable** プロシージャを実行するときと同じです。
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」612 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 フラグを ON に設定する必要はありません。それは、プライマリ・サイトの場合にかぎり必要です。

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 フラグを 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。

この項では、上記のタスクのそれぞれについて説明します。また、Adaptive Server Anywhere LTM の起動方法も説明します。

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 Adaptive Server 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 Adaptive Server 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_owner** 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。

Adaptive Server Anywhere LTM の設定と起動

レプリケーションを実行するには、Adaptive Server 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 Adaptive Server Anywhere LTM を正しく設定してから、Adaptive Server 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 Adaptive Server 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、Adaptive Server Anywhere LTM についての使用情報を検索できます。

```
dbltm -?
```

Adaptive Server 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」27 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、「レプリケーション定義の作成」618 ページで定義したニュース・レプリケーションのためのサブスクリプションを、レプリケート・サイトの **repdb** データベースで作成します。

```
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
```

Adaptive Server 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 Adaptive Server Anywhere データベースは、事前に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 Adaptive Server 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させるには、プライマリ・サイトまたはレプリケート・サイトの Adaptive Server Anywhere データベースごとに Open Server 定義が 1 つずつ必要です。

LTM の設定については、「LTM の設定」629 ページを参照してください。

Replication Server のデータベースの設定

Adaptive Server 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、Adaptive Server Anywhere Replication Agent 製品と共に提供されている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 Adaptive Server 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、Adaptive Server 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、Adaptive Server 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入っています。
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、Adaptive Server 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。Adaptive Server Anywhere は、NULL パスワードを承認しません。

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、Adaptive Server 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. Adaptive Server Anywhere データベース・サーバ上で、Adaptive Server 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 権限のあるユーザとしてそのデータベースに接続します。Adaptive Server Anywhere データベースを作成すると、ユーザ ID **DBA** とパスワード **SQL** を持ち、DBA 権限を付与されたユーザが存在することになります。

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 path¥rssetup.sql
```

path には、設定スクリプトへのパスを入力します。

文字セットと言語の問題

Adaptive Server 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ます。

❖ 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
```

言語は、「言語ラベルの値」435 ページにリストされた言語の 1 つです。

LTM の使用

Adaptive Server Anywhere LTM は Adaptive Server Anywhere トランザクション・ログ内の情報に基づいているので、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」634 ページの項を参照してください。

Adaptive Server Anywhere LTM を Adaptive Server Enterprise LTM の代わりに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。

Adaptive Server 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、Adaptive Server 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

Adaptive Server Anywhere は、**sp_setreplicate** システム・プロシージャをサポートしていません。その代わりに、テーブルが、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、Adaptive Server 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
Date および time 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

Adaptive Server Anywhere は、NULL ではない、長さが 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

Adaptive Server 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。

Adaptive Server 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

Adaptive Server 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 Adaptive Server Anywhere データベースは、関数 APC と他のテーブル APC を組み合わせることができません。

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
```

これらの文には、Adaptive Server Enterprise のプロシージャに **sp_setreplicate** または **sp_setrepproc 'table'** を実行する場合と同じ影響があります。**sp_setrepproc 'function'** 構文はサポートされていません。

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

Adaptive Server 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、Adaptive Server 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

Adaptive Server Anywhere サーバから転送しているか、どの Replication Server に転送するかというような情報です。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」717 ページを参照してください。

設定ファイルの例

- 次は、Adaptive Server Anywhere LTM の設定ファイル例です。

```
# 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。デフォルト設定値は 256 K 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。Adaptive Server Anywhere の文字セット・サポートは、Adaptive Server Enterprise とその他の Open Client/Open Server をベースとしたアプリケーションでの文字セット・サポートとは違う方法で行われています。

この項では、Adaptive Server Anywhere データベースのデータを Replication Server と共有できるように、Adaptive Server Anywhere LTM を設定する方法を説明します。これによって他のデータベースとも共有できるようになります。

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*	ユーティリティと共に使用する、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。

  Adaptive Server 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。

Adaptive Server 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、Adaptive Server 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 Adaptive Server Anywhere LTM の相違の 1 つは、Adaptive Server Enterprise LTM はテンポラリ・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しますが、Adaptive Server Anywhere LTM は古いトランザクション・ログへのアクセスに依存するという点です。Adaptive Server 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、Adaptive Server 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、Adaptive Server 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

Adaptive Server 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」483 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。統合データベースとリモート・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¥YYMMD-Dxx.log* に変更します。**xx** は、**AA** ～ **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

DELETE_OLD_LOGS の Adaptive Server Anywhere データベース・オプションは、デフォルトで OFF に設定されています。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。

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'
```

詳細については、「DELETE_OLD_LOGS オプション [レプリケーション]」 844 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

Adaptive Server 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。

詳細については、「REPLICATE_ALL オプション [レプリケーション]」 891 ページ を参照してください。

LTM の停止

Windows NT/2000/XP のユーザ・インタフェースから LTM を停止させることができます。別の環境では、コマンドを発行して停止させます。

❖ Windows NT/2000/XP で、LTM がサービスとして稼働していないときに停止するには、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
```

第 4 部　ユーティリティ、オプション、プロパティ

この項では、データベース管理ユーティリティ、オプション、プロパティ、制限について説明します。

第 15 章

データベース管理ユーティリティ

この章の内容

Adaptive Server Anywhere には、データベースのバックアップなどの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。この章では、それぞれのデータベース管理ユーティリティについて説明します。

管理ユーティリティの概要

この章では、Adaptive Server Anywhere に付属しているプログラムとデータベース管理ユーティリティに関する参照情報を示します。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。

Sybase Central の包括的なマニュアルについては、Sybase Central のオンライン・ヘルプを参照してください。Sybase Central データベース管理ツールの概要については、『SQL Anywhere Studio の紹介』>「Sybase Central を使用したデータベースの管理」を参照してください。

管理ユーティリティでは、一連のシステム環境変数を使用します。これらの変数については、「環境変数」363 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。これらの文は、『ASA SQL リファレンス・マニュアル』>「SQL 文」にリストされています。

設定ファイルの使用

SQL Anywhere Studio 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。

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。

設定ファイルに含まれる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、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 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 user ASA with
password SQL
 -c "uid=DBA;pwd=SQL;dbf=c:¥Program Files¥Sybase¥SQL
Anywhere 9¥asademo.db"
 #Perform a full check on each table
 -f
 #Log output messages to the specified file
 -o "c:¥validationlog.txt
```

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
```

Adaptive Server Anywhere コンソール・ユーティリティ

Adaptive Server Anywhere コンソールは、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。

Adaptive Server Anywhere コンソールは、Windows CE、AIX、HP-UX、HP-UX Itanium、Linux Itanium、Compaq Tru64 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、設定レベル、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、Adaptive Server 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 95/98/Me、Windows NT/2000/XP、Mac OS X、Linux など) を実行するマシン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、「データベースのパフォーマンスと接続プロパティ」913 ページを参照してください。

dbconsole ユーティリティを使用した接続のモニタリング

構文

dbconsole [*options*]

オプション	説明
-c *"keyword=value; ..."*	データベース接続パラメータ

説明

Adaptive Server Anywhere コンソールでは、クライアント・マシンからサーバをモニタできます。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。Adaptive Server Anywhere コンソールは、複数の接続に対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. Adaptive Server 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。

   Adaptive Server 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] ー [オプション] を選択するとアクセスできます。

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*。

コンソール・ユーティリティのオプション

-c "keyword=value; …"　接続パラメータを指定します。

サポートされる接続パラメータの詳細については、「接続パラメータ」 236 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

バックアップ・ユーティリティを使って、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・ログをバックアップできます。

次の方法で、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。
- コマンド・プロンプトで dbbackup コマンドを入力する。バックアップ・プロシージャをバッチまたはコマンド・ファイルへ組み込む場合には、このユーティリティが便利です。
- Interactive SQL の BACKUP DATABASE 文を使用する。

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。

実行中のデータベースに対してバックアップ・ユーティリティを実行するのは、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。他のアプリケーションまたはユーザがデータベースを使っている間に、データベースをバックアップできます。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

推奨されるバックアップ手順については、「バックアップとデータ・リカバリ」483 ページを参照してください。

[バックアップ・イメージ作成] ウィザードを使用したデータベースのバックアップ

❖ データベース・ファイルまたは実行中のデータベースをバックアップするには、次の手順に従います。

1. 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。
2. 右ウィンドウ枠にある [ユーティリティ] タブをクリックします。
3. 右ウィンドウ枠の [バックアップ・イメージの作成] をダブルクリックします。

   [バックアップ・イメージ作成] ウィザードが表示されます。
4. ウィザードの指示に従います。

Sybase Central からデータベースをバックアップする方法については、「バックアップとデータ・リカバリ」483 ページを参照してください。

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。

- [ツール] ー [Adaptive Server Anywhere 9] ー [バックアップ・イメージの作成] を選択する。
- 左ウィンドウ枠でデータベースを選択し、[ファイル] ー [バックアップ・イメージの作成] 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 [バックアップ・イメージの作成] を選択する。

---

dbbackup コマンド・ライン・ユーティリティを使用したデータベースのバックアップ

構文

dbbackup [*options*] *target-directory*

オプション	説明
@data	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出す
-c *"keyword=value; …"*	データベース接続パラメータを指定する
-d	メイン・データベース・ファイルだけをバックアップする
-l *file*	トランザクション・ログをファイルにライブ・バックアップする
-n	バックアップ・トランザクション・ログ用に命名規則を変更する
-o *filename*	ファイルに出力メッセージのログを取る
-q	クワイエット・モード (メッセージを表示しない)
-r	トランザクション・ログの名前を変更し再起動する
-s	BACKUP 文を使用してサーバでイメージのバックアップを実行する
-t	トランザクション・ログだけをバックアップする
-w	ライト・ファイルだけをバックアップする
-x	トランザクション・ログを削除し再起動する
-xo	バックアップせずにトランザクション・ログを削除し、再起動する
-y	確認メッセージを表示せずにファイルを置き換える
target-directory	バックアップ・ファイルのコピー先ディレクトリ

説明

-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。

-s が指定されている場合、BACKUP 文を使用してサーバにバックアップが作成されます。指定しない場合、バックアップはクライアント・マシン上に作成されます。

バックアップ・ユーティリティのオプション

@data　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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

接続パラメータ (-c)　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。DBA 権限または REMOTE DBA 権限 (SQL Remote) のあるユーザ ID を使用してください。

接続パラメータの詳細については、「接続パラメータ」236 ページを参照してください。

たとえば、次のコマンドは **sample_server** サーバ上で実行している **asademo** データベースのバックアップを取り、パスワード **SQL** のユーザ ID **DBA** として *asabackup* ディレクトリに接続します。

```
dbbackup -c
"eng=sample_server;dbn=asademo;uid=DBA;pwd=SQL"
asabackup
```

メイン・データベースのみバックアップする (-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。

ライブ・バックアップする (-l (小文字の L)) 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。

-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。

ライブ・バックアップの詳細については、「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」509 ページと「ライブ・バックアップの作成」539 ページを参照してください。

バックアップ・トランザクション・ログの命名規則を変更する (-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 オプションを使用して、ログ・コピーの名前を変更できます。例を示します。

```
dbbackup -x -n
```

ファイルに出力メッセージのログを取る (-o) 指定したファイルに、出力メッセージを書き込みます。

クワイエット・モードで処理を実行する (-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。

トランザクション・ログの名前を変更し、新しいログを起動する (-r)　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。

サーバでイメージのバックアップを実行する (-s)　このオプションにより、BACKUP 文を使用してサーバで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たい場合は、-s または -y を指定しないでください。

トランザクション・ログ・ファイルのみバックアップする (-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。

データベース・ライト・ファイルのみバックアップする (-w)　データベース・ライト・ファイルの詳細については、「ライト・ファイル・ユーティリティ (旧式)」789 ページ を参照してください。

トランザクション・ログを削除して再起動する (-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。

バックアップせずにトランザクション・ログを削除して再起動する (-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。この操作により、レプリケーションではない環境でディスク領域を解放できます。

確認メッセージを表示することなく処理を実行する (-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たい場合は、-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。

照合ユーティリティ

照合ユーティリティを使って、照合 (ソート順序) を抽出し、カスタム照合を使ったデータベースの作成に適切なファイルへ入れることができます。

生成されたファイルを修正し、Sybase Central で使用するか、dbinit の -z オプションを使用してカスタム照合を使用する新規データベースを作成できます。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

カスタム照合ファイルの照合ラベルを変更します。そうしないと、カスタム照合は、その基となるオリジナルの照合と競合します。

```
Collation label (name)
```

カスタム照合順の詳細については、「カスタム照合を使用してデータベースを作成する」467 ページを参照してください。

次の方法で、照合ユーティリティにアクセスできます。

- Sybase Central の [カスタム照合作成] ウィザードを使用する。
- コマンド・プロンプトで、dbcollat コマンドを入力する。

[カスタム照合作成] ウィザードを使用した照合の抽出

❖ **[カスタム照合作成] ウィザードを使用するには、次の手順に従います。**

1. 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。
2. 右ウィンドウ枠にある [ユーティリティ] タブをクリックします。
3. 右ウィンドウ枠の [カスタム照合の作成] をダブルクリックします。

[カスタム照合作成] 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
Sybase Central では、次の方法で [カスタム照合作成] ウィザードを利用することもできます。

- [ツール] ー [Adaptive Server Anywhere 9] ー [カスタム照合の作成] を選択する。
- 左ウィンドウ枠でデータベースを選択し、[ファイル] ー [カスタム照合の作成] 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 [カスタム照合の作成] を選択する。

dbcollat コマンド・ライン・ユーティリティを使用した照合の抽出

構文 **dbcollat** [*options*] *output-file*

オプション	説明
@data	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出す
-c *"keyword=value; …"*	データベース接続パラメータを指定する
-d *collation-file*	定義ファイルを INSERT 文に変換する。照合マッピングは *output-file* に配置されます。
-e	空のマッピングを含める
-o *filename*	ファイルに出力メッセージのログを取る
-q	クワイエット・モード (メッセージを表示しない)
-x	拡張文字 (7F-FF) に 16 進数を使用する

オプション	説明
-y	確認メッセージを表示せずにファイルを置き換える
-z *col-seq*	照合順ラベルを指定する

照合ユーティリティのオプション

@data　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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

接続パラメータ (-c)　接続パラメータの詳細については、「接続パラメータ」236 ページを参照してください。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。

たとえば、次のコマンドは **sample_server** サーバで実行される **asademo** データベースから照合ファイルを抽出して、パスワードが **SQL** のユーザ ID **DBA** として接続します。

```
dbcollat -c
"eng=sample_server;dbn=asademo;uid=DBA;pwd=SQL"
c:¥sample¥col
```

照合定義ファイルの照合を記述する INSERT 文へ変換する (-d)　データベースを作成すると、照合は SYS.SYSCOLLATION システム・テーブルに挿入されます。照合から文字セットへのマッピングと Sybase TDS 照合も SYS.SYSCOLLATIONMAPPINGS システム・テーブルに挿入されます。この照合は、*collseqs.sql* ファイルに用意されている照合

のセットまたは *custom.sql* ファイルのカスタム照合から選択されます。これらのファイルは、Adaptive Server Anywhere 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。

SYSCOLLATIONMAPPINGS システム・テーブルの詳細については、『ASA SQL リファレンス・マニュアル』>「SYSCOLLATIONMAPPINGS システム・テーブル」を参照してください。

カスタム照合は *custom.sql* スクリプトに追加されます。−d オプションは、指定した照合定義ファイルを *custom.sql* にコピー可能な INSERT 文に変換します。

たとえば、次のように dbcollat コマンドの -d オプションを使用できます。

```
dbcollat -d coll-defn-file custom-file
```

ファイル *coll-defn-file* は、照合定義として読み込み、解析されます。出力は、*custom-file* に書き込まれます。*custom-file* の内容は、必ず *custom.sql* に追加されます。

−d オプションを使用したカスタム照合の作成については、「カスタム照合の作成」465 ページを参照してください。

空のマッピングを含める (-e) 通常、照合は文字がソートする実際の値を指定しません。代わりに、照合の各行が前の行より 1 つ上の位置へソートを行います。ただし、古い照合には、いくつかのソート位置の間にギャップがあります。通常、照合ユーティリティはこのギャップを省略し、明示的なソート位置にある次の行を書き込みます。このオプションを使うと、照合ユーティリティはギャップの中の各行に対し、空のマッピング (コロン (:) だけで構成される) を書き込みます。

ファイルに出力メッセージのログを取る (-o) 指定したファイルに、出力メッセージを書き込みます。

クワイエット・モードで処理を実行する (-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。

拡張文字で 16 進数を使用する [7F から FF] (-x)　拡張シングルバイト文字 (16 進数 7F より大きい値) は画面に正確に表示されることもあれば、表示されないこともあります。これは、コンピュータで使用しているコード・ページが、抽出された照合で使用されているコード・ページと同じかどうかによって決まります。このオプションを使うと、照合ユーティリティは、16 進数 7F またはそれ以上のすべての文字を ¥x*dd* のフォームで、2 バイトの 16 進数文字として書き込みます。次に例を示します。

```
¥x80, ¥xFE
```

-x オプションを指定しなかった場合は、¥x00 ～ ¥x1F、¥x7F、¥xFF に該当する文字だけが 16 進数の形式で書き込まれます。

確認メッセージを表示することなく処理を実行する (-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、既存の照合ファイルが自動的に置き換えられます。

照合順ラベルを指定する (-z)　抽出される照合のラベルを指定します。推奨される照合順の名前を確認するには、次のコマンドを実行します。

```
dbinit -l
```

使用可能な照合ラベルに -z オプションを指定すると、dbcollat はデータベースに接続しません。このオプションを指定しないと、データベースに接続し、そのデータベースが使用している照合ラベルを抽出します。照合ラベルとデータベースの照合ラベルが一致しない場合は、エラーが返されます。

圧縮ユーティリティ (旧式)

圧縮ユーティリティを使って、データベース・ファイルを圧縮できます。圧縮ユーティリティは、指定されたデータベース・ファイルからその圧縮データベース・ファイルを作成します。圧縮データベース・ファイルは、通常、元のサイズの 40 ～ 60% の大きさになります。
データベース・サーバは圧縮データベース・ファイルを更新できません。ライト・ファイルとともに、読み込み専用ファイルとして使用します。

推奨されなくなった機能
圧縮データベースは推奨されなくなりました。

圧縮ユーティリティは、メイン・データベース・ファイル以外のファイルを圧縮できません。圧縮するデータベースは、実行中であってはなりません。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

次の方法で、圧縮ユーティリティにアクセスします。

- Sybase Central の [データベース圧縮] ウィザードを使用する。
- コマンド・プロンプトで、dbshrink 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。

警告
暗号化されたデータベースを圧縮すると、データベースから暗号化が削除されます。

[データベース圧縮] ウィザードを使用したデータベースの圧縮

❖ データベース・ファイルを圧縮するには、次の手順に従います。

1　左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。

2　右ウィンドウ枠にある [ユーティリティ] タブをクリックします。

3　右ウィンドウ枠にある [データベースの圧縮] をダブルクリックします。

[データベース圧縮] 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

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 [データベースの圧縮] を選択するか、以下のいずれかのオプションを選択して、Sybase Central の [データベース圧縮] ウィザードを使用することもできます。

- [ツール] － [Adaptive Server Anywhere 9] － [データベースの圧縮] を選択する。
- 左ウィンドウ枠でデータベースを選択し、[ファイル] － [データベースの圧縮] 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 [データベースの圧縮] を選択する。

dbshrink コマンド・ライン・ユーティリティを使用したデータベースの圧縮

構文　**dbshrink** [*options*] *database-file* [*compressed-database-file*]

オプション	説明
@data	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出す
-ek *key*	暗号化キーを指定する
-ep	暗号化キーを入力するよう要求する
-o *filename*	ファイルに出力メッセージのログを取る
-q	クワイエット・モード (メッセージを表示しない)
-y	確認メッセージを表示せずに既存の出力ファイルを置き換える

説明

dbshrink ユーティリティは *database-file* を読み込み、圧縮データベース・ファイルを作成します。デフォルトでは、圧縮ファイル名は元のファイル名と同じ名前ですが、拡張子 *.cdb* が付きます。出力ファイル名 (拡張子付き) は、入力ファイル名 (拡張子付き) と同じにしないでください。

圧縮ユーティリティのオプション

@data 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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

暗号化キーを指定する (-ek) 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。強力に暗号化されたデータベースの場合、-ek または -ep をどちらか指定します。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。

暗号化キーを入力するよう要求する (-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。強力に暗号化されたデータベースの場合、-ek または -ep をどちらか指定します。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。

ファイルに出力メッセージのログを取る (-o)　指定したファイルに、出力メッセージを書き込みます。

クワイエット・モードで処理を実行する (-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。

確認メッセージを表示することなく処理を実行する (-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、既存の圧縮データベース・ファイルが自動的に置き換えられます。

データ・ソース・ユーティリティ

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 Adaptive Server Anywhere ODBC データ・ソースの作成、削除、記述、リストを実行します。

ODBC アドミニストレータを使用したデータ・ソースの作成については、「ODBC データ・ソースの使用」70 ページを参照してください。

dbdsn コマンド・ライン・ユーティリティを使用した ODBC データ・ソースの管理

構文

dbdsn [*modifier-options*]
{ **-l**[**u** | **s**] [**-qq**]
| **-d**[**u** | **s**] *dsn*
| **-g**[**u** | **s**] *dsn*
| **-w**[**u** | **s**] *dsn* [*details-options*;…]
| **-cl**[**-qq**] }

パラメータ

主要オプション	説明
@data	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出す
-l[**u** | **s**] [**qq**]	すべての Adaptive Server Anywhere ユーザまたはシステムのデータ・ソースをリストする。デフォルトは、ユーザのデータ・ソース。このオプションと一緒に -qq を使用すると、バナーまたはタイトルなしで DSN がリストされます。
-d[**u** | **s**] *dsn*	指定した Adaptive Server Anywhere ユーザまたはシステムのデータ・ソースを削除する。デフォルトは、ユーザのデータ・ソース。
-g[**u** | **s**] *dsn*	指定した Adaptive Server Anywhere ユーザまたはシステムのデータ・ソースの詳細をリスト (取得) する。デフォルトは、ユーザのデータ・ソース。

主要オプション	説明
-w[**u** | **s**] *dsn* [*details-options*]	ユーザまたはシステムのデータ・ソース定義を作成する (書き込む)。デフォルトは、ユーザのデータ・ソース。
-cl[**-qq**]	使用可能な接続パラメータをリストする。このオプションと一緒に -qq を使用すると、バナー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。

変更子オプション	説明
-b	簡易。データ・ソースの接続文字列を表示します。
-cm	データ・ソースの作成コマンドを表示する。
-q	クワイエット。バナーを表示しません。
-v	冗長。表形式で接続パラメータを表示します。
-y	確認メッセージを表示せずにデータ・ソースを削除または上書きする

詳細オプション	説明
-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する。DBF の値が絶対ファイル名でない場合、データ・ソース・ユーティリティは現在の作業ディレクトリ (CWD) を付加します。
-c "*keyword***=***value***;…"**	データベース接続パラメータを指定する
-ec *encryption type*	すべてのネットワーク・パケットを暗号化する
-o *filename*	クライアント・メッセージを *filename* に書き込む
-p *size*	ネットワーク・パケットの最大サイズを設定する

詳細オプション	説明
-r	複数レコード・フェッチを無効にする
-tl *seconds*	クライアントの活性タイムアウト時間を設定する
-x *list*	実行するネットワーク・ドライバをリストする
-z	デバッグ情報を表示する
server-name	指定のデータベース・サーバに接続する

参照

- 「ODBC データ・ソースの使用」70 ページ
- 「UNIX での ODBC データ・ソースの使用」74 ページ

説明

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 Adaptive Server 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。

UNIX オペレーティング・システムでは、データ・ソースは *odbc.ini* ファイルに保存されます。データ・ソース・ユーティリティを使用して Adaptive Server Anywhere ODBC データ・ソースを UNIX 上で作成または削除する場合、ユーティリティは自動的に *.odbc.ini* ファイルの [ODBC データ・ソース] セクションを更新します。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 ASANY9 環境変数の設定に基づき Adaptive Server 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。

.odbc.ini ファイルの詳細については、『Mobile Link およびリモート・データ・アクセスの ODBC ドライバ』> 「システム情報ファイル (.odbc.ini)」を参照してください。

警告

Adaptive Server 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ようにしてください。その他の

データ・ソース (Mobile Link 同期など) を使用する場合は、*odbc.ini* ファイルを難読化すると、その他のドライバが正しく機能しなくなる場合があります。

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。

データ・ソース・ユーティリティ オプション

データ・ソース・ユーティリティを使用するときは、主要オプション、変更子オプション、詳細オプションを選択できます。

主要オプション

@data　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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

使用可能な接続パラメータをリストする (-cl)　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。

データ・ソース・ユーティリティ (dbdsn) によってサポートされる Adaptive Server Anywhere 接続パラメータの詳細については、「接続パラメータ」236 ページを参照してください。

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。

名前	説明
Driver	このパラメータにより、接続の ODBC ドライバを次のように指定できます。`Driver=<`*driver-name*`>`。デフォルトで使用されるドライバは **Adaptive Server Anywhere 9.0** です。*driver-name* は、Adaptive Server Anywhere *X*.0 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 **Adaptive Server 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 ASANY9 環境変数の設定に基づき Adaptive Server 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。
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。

名前	説明
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** これは逐次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。 詳細については、『ASA SQL ユーザーズ・ガイド』>「独立性レベルの選択」を参照してください。
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。

名前	説明
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	Adaptive Server 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 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。

指定のデータ・ソースを削除する (-d)　指定の Adaptive Server Anywhere データ・ソースを削除します。**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。-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。

指定のデータ・ソースの詳細をリスト (取得) する (-g) 指定された Adaptive Server 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。**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。

定義されたデータ・ソースをリストする (-l) 使用可能な Adaptive Server 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。**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。

データ・ソース定義を作成する (書き込む) (-w) 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。**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。-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。

変更子オプション

データ・ソースの接続文字列を表示する (-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。

データ・ソースの作成コマンドを表示する (-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マシンにデータ・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、マシンに 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"
```

バナーを表示しない (-q)　情報バナーを表示しません。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。

バナーおよびタイトルを表示しない (-qq)　情報バナーとタイトルの両方を表示しません。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。

表形式で接続パラメータを表示する (-v)　数行のリスト出力を表としてフォーマットします。

確認メッセージを表示せずにデータ・ソースを削除または上書きする (-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

接続パラメータ (-c)　接続パラメータを接続文字列として指定します。

詳細については、「接続パラメータ」236 ページを参照してください。

絶対ファイル名 (-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 (Windowx 9x) 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。

ネットワーク・パケットを暗号化する (-ec)　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。

詳細については、「Encryption 接続パラメータ [ENC]」256 ページ を参照してください。

ファイルに出力メッセージのログを取る (-o)　指定したファイルに、出力メッセージを書き込みます。デフォルトでは、メッセージはコンソールに出力されます。

詳細については、「Logfile 接続パラメータ [LOG]」268 ページ を参照してください。

ネットワーク・パケットの最大サイズを設定する (-p) ネットワーク通信で使用するパケットの最大サイズをバイト単位で指定します。300 より大きく 16000 より小さい値を指定してください。デフォルト設定は 1460 です。

詳細については、「CommBufferSize 接続パラメータ [CBSIZE]」241 ページ を参照してください。

複数レコード・フェッチを無効にする (-r) 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。このオプションを使用することで、そのような動作を無効にできます。

詳細については、「DisableMultiRowFetch 接続パラメータ [DMRF]」255 ページ を参照してください。

クライアントの活性タイムアウトを設定する (-tl) 使用されなくなった接続を終了します。値は秒数で指定します。

デフォルトは、サーバ設定です。サーバ設定のデフォルトは、120 秒です。

詳細については、「LivenessTimeout 接続パラメータ [LTO]」267 ページ を参照してください。

通信リンクを設定する (-x) 実行するネットワーク・ドライバのリストをカンマで区切って指定します。

詳細については、「CommLinks 接続パラメータ [LINKS]」243 ページ を参照してください。

デバッグ情報を表示する (-z) 起動時に通信リンクに関する診断情報を提供します。

server-name 指定のサーバに接続します。最初の 40 文字が使用されます。

詳細については、「データベース・サーバ」153 ページを参照してください。

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" -v
```

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" -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、既存の定義を上書きします。

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 (バナーとタイトルなし)。

```
dbdsn -l -qq
```

すべての接続パラメータをリストします (バナーとタイトルなし)。

```
dbdsn -cl -qq
```

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=E:¥asa90¥my.db* が含まれます。

```
E:¥asa90> dbdsn -w testdsn -cw -c
uid=dba;pwd=sql;eng=asa;dbf=my.db
```

ASA 9.0 サンプル・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 "ASA 9.0 Sample" > restoredsn.bat
```

restoredsn.bat ファイルには以下が含まれています。

```
dbdsn -y -wu "ASA 9.0 Sample" -c "UID=dba;PWD=sql;
 DBF=C:¥Program Files¥Sybase¥SQL Anywhere 9¥asademo.db;
 ENG=asademo9;START=C:¥Program Files¥Sybase¥SQL
Anywhere 9¥
 win32¥dbeng9.exe -c 8m;ASTOP=yes;
 Description=Adaptive Server Anywhere Sample Database"
```

消去ユーティリティ

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。

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。

次の方法で、消去ユーティリティにアクセスできます。

- Sybase Central の [データベース消去] ウィザードを使用する。
- コマンド・プロンプトで、dberase 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。

詳細については、『ASA SQL リファレンス・マニュアル』> 「DROP DATABASE 文」を参照してください。

[データベース消去] ウィザードを使用したデータベースの消去

❖ **データベース・ファイルを消去するには、次の手順に従います。**

1 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。

2 右ウィンドウ枠にある [ユーティリティ] タブをクリックします。

3 右ウィンドウ枠にある [データベースの消去] をダブルクリックします。

[データベース消去] 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。

Sybase Central からデータベースを消去する方法については、『ASA SQL ユーザーズ・ガイド』> 「データベースの消去」を参照してください。

ヒント

Sybase Central では、次の方法で [データベース消去] ウィザードを利用することもできます。

- [ツール] － [Adaptive Server Anywhere 9] － [データベースの消去] を選択する。
- 左ウィンドウ枠でサーバを選択し、[ファイル] － [データベースの消去] を選択する。
- サーバを右クリックし、ポップアップ・メニューから [データベースの消去] を選択する。

dberase コマンド・ライン・ユーティリティを使用したデータベースの消去

構文

dberase [*options*] *database-file*

オプション	説明
@data	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出す
-ek *key*	暗号化キーを指定する
-ep	暗号化キーを入力するよう要求する
-o *filename*	ファイルに出力メッセージのログを取る
-q	クワイエット・モードで作動する (メッセージを表示しない)
-y	確認メッセージを表示せずにファイルを消去する

説明

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。

消去ユーティリティのオプション

@data　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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

暗号化キーを指定する (-ek) 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。強力に暗号化されたデータベースの場合、-ek または -ep をどちらか指定します。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。

暗号化キーを入力するよう要求する (-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。強力に暗号化されたデータベースの場合、-ek または -ep をどちらか指定します。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。

ファイルに出力メッセージのログを取る (-o) 指定したファイルに、出力メッセージを書き込みます。

クワイエット・モードで処理を実行する (-q) 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。

確認メッセージを表示することなく処理を実行する (-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、各ファイルが自動的に削除されます。-q を指定する場合、-y も指定しないと操作は失敗します。

ファイル非表示ユーティリティ

ファイル非表示ユーティリティを使用して、各ファイルの内容を隠すために単純暗号化を設定ファイルと初期化ファイルに追加できます。

dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す

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
encrypted-configuration-file	難読化された新しいファイルの名前

説明

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 Adaptive Server 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いてください。

.ini ファイルの内容の非表示

多くの場合、Adaptive Server 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 (*asaldap.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。

❖ **.ini 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 asaldap.ini asaldap.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 asaldap.ini.org asaldap.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 *asaldap.ini.org* ファイルを保護するか、安全な場所に保存します。

asaldap.ini ファイルに変更を加えるには、*asaldap.ini.org* ファイルを編集し、手順 2 を繰り返します。

警告

Adaptive Server 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に単純暗号化を追加しないようにしてください。その他のデータ・ソース (Mobile Link 同期など) を使用する場合は、*.odbc.ini* ファイルの内容を読みにくくすると、その他のドライバが正しく機能しなくなる場合があります。

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
 path¥asademo.db
```

(先頭に # がある行はコメントとして処理されます。)

ファイルの名前は *sample.txt* とします。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。

```
dbeng9 @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。

```
dbsrv9 @encrypted_sample.txt
```

暗号化の詳細については、『SQL Anywhere Studio セキュリティ・ガイド』> 「安全なデータの管理」を参照してください。

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」12 ページを参照してください。

次のコマンドは、*asaldap.ini* ファイルに単純暗号化を追加します。

```
dbfhide asaldap.ini encrypted_asaldap.ini
```

ヒストグラム・ユーティリティ

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。このユーティリティは、Windows で動作する Excel 97 以降でのみ有効です。

dbhist コマンド・ライン・ユーティリティを使用したヒストグラムの変換

構文

dbhist [*options*] **-t** *table-name* [*excel-output-name*]

パラメータ

オプション	説明
@data	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出す
-c *options*	接続文字列
-n *colname*	ヒストグラムに関連付けるカラム
-t *table-name*	ヒストグラムを生成するテーブルの名前
-u *owner*	テーブル所有者
excel-output-name:	生成される Excel ファイルの名前

説明

ヒストグラムは SYSCOLSTAT 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。このユーティリティは、Windows で動作する Excel 97 以降でのみ有効です。

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。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

sa_get_histogram ストアド・プロシージャの詳細については、『ASA SQL リファレンス・マニュアル』> 「sa_get_histogram システム・プロシージャ」を参照してください。

ヒストグラム・ユーティリティのオプション

@data　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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

接続文字列を指定する (-c)　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」236 ページを参照してください。

カラムを指定する (-n)　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。

テーブル名を指定する (-t)　ヒストグラムを生成するテーブルの名前を指定します。

所有者を指定する (-u)　テーブル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。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。

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。次の文 (すべて同一行に入力) は、データベース *asademo.db* にあるテーブル sales_order_items のカラム prod_id 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=asademo.db"
       -n prod_id -t sales_order_items histgram.xls
```

次の文は、テーブル sales_order で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。asademo はすでにロードされているものとします。この文は、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 sales_order -c "UID=DBA;PWD=SQL"
```

情報ユーティリティ

情報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を表示できます。

dbinfo コマンド・ライン・ユーティリティを使用したデータベース情報の取得

構文

dbinfo [*options*]

オプション	説明
@data	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出す
-c *"keyword=value; ..."*	データベース接続パラメータ
-o *filename*	ファイルに出力メッセージのログを取る
-q	クワイエット・モードで作動する
-u	ページの使用状況に関する統計情報を出力する

説明

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。データベースの作成日時、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、インストールされている Java クラスのバージョン、その他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。

情報ユーティリティのオプション

@data　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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

接続パラメータ (-c) 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」236 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。

ファイルに出力メッセージのログを取る (-o) 指定したファイルに、出力メッセージを書き込みます。

クワイエット・モードで処理を実行する (-q) 出力メッセージを表示しません。

ページの使用状況に関する統計情報 (-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。

初期化ユーティリティ

初期化ユーティリティを使って、データベースを初期化 (作成) できます。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
- パスワードに対する大文字と小文字の区別の有無
- 暗号化ファイルとしての格納
- 比較における後続ブランクの処理
- ページ・サイズ
- 照合順

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・ユーティリティを使用して、後で変更できます。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

次の方法で、初期化ユーティリティにアクセスできます。

- Sybase Central の [データベース作成] ウィザードを使用する。
- Interactive SQL から CREATE DATABASE 文を使用する。
- コマンド・プロンプトで、dbinit 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。

データベースの作成については、『ASA SQL リファレンス・マニュアル』> 「CREATE DATABASE 文」を参照してください。

[データベース作成] ウィザードを使用したデータベースの作成

❖ **データベースを作成するには、次の手順に従います。**

1. 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。
2. 右ウィンドウ枠にある [ユーティリティ] タブをクリックします。
3. 右ウィンドウ枠で、[データベースの作成] をダブルクリックします。

   [データベース作成] 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

Sybase Central では、次の方法で [データベース作成] ウィザードを利用することもできます。

- [ツール] － [Adaptive Server Anywhere 9] － [データベースの作成] を選択する。
- 左ウィンドウ枠でサーバを選択し、[ファイル] － [データベースの作成] を選択する。
- サーバを右クリックし、ポップアップ・メニューから [データベースの作成] を選択する。

Sybase Central でデータベースを作成する方法については、『ASA SQL ユーザーズ・ガイド』> 「データベースの作成」を参照してください。

dbinit コマンド・ライン・ユーティリティを使用したデータベースの作成

構文 **dbinit** [*options*] *new-database-file*

オプション	説明
@data	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出す
-b	比較およびフェッチ用に文字列の埋め込みをブランクにする
-c	すべての文字列比較を行うときに大文字と小文字を区別する
-cp	パスワードの大文字と小文字の区別
-e	単純暗号を使用してデータベースを暗号化する
-ea *algorithm*	データベースの暗号化に使用する強力な暗号化アルゴリズムを指定する。AES か AES_FIPS を選択できます。
-ek *key*	強力な暗号化を行うための暗号キーを指定する
-ep	強力な暗号化を行うための暗号キーを入力するためのプロンプトを表示する
-i	Sybase jConnect サポートをインストールしない
-ja	デフォルトのランタイム Java クラスをインストールする
-jdk *version*	Java Development Kit の指定したバージョンのエントリをインストールする
-k	Watcom SQL 互換のビュー SYS.SYSCOLUMNS と SYS.SYSINDEXES を省略する
-l	推奨される照合順を表示する
-m *file-name*	トランザクション・ログ・ミラーを使用する (デフォルトはミラーなし)
-n	トランザクション・ログなし
-o *filename*	ファイルに出力メッセージのログを取る
-p *page-size*	ページ・サイズを設定する
-q	クワイエット・モード (メッセージを表示しない)
-s	ディスクにページを書き込むときにチェックサムを使用する

オプション	説明
-t *log-name*	トランザクション・ログ・ファイル名 (デフォルトの設定は、拡張子 *.log* の付いたデータベース名)
-z *col-seq*	比較に使用する照合順

説明

たとえば、次のようにして 4096 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 4096 test.db
```

初期化ユーティリティのオプション

@data 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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

ブランク埋め込み (-b) 比較するために後続ブランクを無視し、文字配列にフェッチされた文字列を埋め込みます。たとえば、次の 2 つの文字列について考えます。

```
'Smith'
 'Smith   '
```

これらの文字列は、ブランク埋め込みで作成されたデータベースでは等しくなります。

このオプションは ISO/ANSI SQL 規格と互換性を保つために用意されています。この規格では、比較をするときに後続ブランクが無視されます。デフォルトでは、ブランクは比較の際に意味を持ちます。

すべての文字列の比較で大文字と小文字を区別する (-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。

パスワードの大文字と小文字の区別 (-cp)　デフォルトでは、パスワードの大文字と小文字の区別は、データベースの大文字と小文字の区別と同じです。ただし、-cp オプションを指定する場合、データベースでの設定にかかわらず、パスワードは大文字と小文字が区別されます。パスワードで大文字と小文字を区別するように指定するには、**-cp-** を指定します。

ユーザ ID とパスワード
すべてのデータベースは、少なくとも 1 つのユーザ ID **DBA** (パスワード **SQL**) が作成されます。パスワードで使用される拡張文字は、データベースの設定にかかわらず、大文字小文字を区別します。大文字と小文字を区別するデータベースであっても、データベース内のユーザ ID およびその他の識別子について大文字と小文字は区別されません。

単純暗号化を使用してデータベースを暗号化する (-e)　単純暗号化を使用すると、他のユーザがディスク・ユーティリティを使用してファイルを参照し、データベースのデータを解読することをより困難にします。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。

この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。セキュリティを強化するには、-ea と -ek オプションを指定して強力な暗号化を使用できます。

暗号化アルゴリズムを指定する (-ea)　このオプションにより、新しいデータベースの暗号化に強力な暗号化アルゴリズムを選択できます。AES (デフォルト) または AES_FIPS (FIPS 承認のアルゴリズムの場合)

を選択できます。AESS_FIPS は個別のライブラリを使用するため、AES との互換性はありません。AES_FIPS はサポートされるすべての 32 ビット Windows プラットフォームでのみ使用できます。アルゴリズム名には大文字と小文字の区別はありません。-ea 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。

詳細については、『SQL Anywhere Studio セキュリティ・ガイド』> 「高度な暗号化」を参照してください。

別途ライセンスを取得できるオプションが必要
AES_FIPS を使用した強力なデータベース暗号化には、個別にライセンスを取得可能な SQL Anywhere Studio セキュリティ・オプションを入手する必要があります。このセキュリティ・オプションは、輸出規制対象品目です。

このコンポーネントを注文するには、『SQL Anywhere Studio の紹介』> 「別途ライセンスが入手可能なコンポーネント」を参照してください。

暗号化キーを指定する (-ek) 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。-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。

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。

次のようなものはデータベース暗号化キーに使用できません。

- 空白スペースまたは一重引用符か二重引用符で始まるキー
- 空白スペースで終わるキー
- セミコロンを含むキー

詳細については、『SQL Anywhere Studio セキュリティ・ガイド』> 「高度な暗号化」を参照してください。

暗号化キーを入力するよう要求する (-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。

詳細については、『SQL Anywhere Studio セキュリティ・ガイド』> 「高度な暗号化」を参照してください。

Sybase jConnect サポートをインストールしない (-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 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、『ASA プログラミング・ガイド』> 「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」を参照してください。

ランタイム Java クラスをインストールする (-ja)　データベースで Java を使用するには、ランタイム Java クラスをインストールします。-ja を指定すると、バージョン 1.3 の JDK がデータベースにインストールされます。

たとえば、次のコマンドは、データベース内で JDK 1.3 アプリケーションをサポートするデータベースを作成します。

```
dbinit -ja java2.db
```

ランタイム・クラスをインストールするとデータベースのサイズが数メガバイト増加するため、Java クラスを使う予定がない場合は、-ja オプションを省略してインストールを避けることができます。

[データベース・アップグレード] ウィザードまたは ALTER DATABASE 文を使用すれば、後でランタイム Java クラスを追加できます。

詳細については、『ASA プログラミング・ガイド』> 「データベースを Java 実行可能にする」と『ASA SQL リファレンス・マニュアル』> 「ALTER DATABASE 文」を参照してください。

JDK の指定バージョンのサポートをインストールする (-jdk) 1.3 以外のバージョンの Java をデータベースにインストールする場合は、-jdk オプションの後にバージョン番号を指定します。現在、サポートされている Java のその他のバージョンは 1.1.8 のみです。

たとえば、次のコマンドは、データベース内で JDK 1.1.8 アプリケーションをサポートするデータベースを作成します。

```
dbinit -jdk 1.1.8 java2.db
```

ランタイム・クラスをインストールするとデータベースのサイズが数メガバイト増加するため、Java クラスを使う予定がない場合は、-jdk オプションを省略してインストールを避けることができます。

-jdk オプションは、-ja オプションも暗黙で指定したことになります。

[データベース・アップグレード] ウィザードまたは ALTER DATABASE 文を使用すれば、後でランタイム Java クラスを追加できます。

詳細については、『ASA プログラミング・ガイド』> 「データベースを Java 実行可能にする」と『ASA SQL リファレンス・マニュアル』> 「ALTER DATABASE 文」を参照してください。

Watcom SQL との互換性ビューを表示しない (-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。

使用可能な照合順をリストする (-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。

トランザクション・ログ・ミラーを使用する (-m)　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、Adaptive Server 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。

トランザクション・ログを使用しない (-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。

ファイルに出力メッセージのログを取る (-o)　指定したファイルに、出力メッセージを書き込みます。

ページ・サイズ (-p)　データベースのページ・サイズには、1024、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 2048 バイトです。サイズにこれ以外の値を指定すると、その値より大きい次の値に変更されます。

大規模なデータベースでは通常、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。また、インデックスのレベルの数は、それぞれのページに保存されるエントリが増えるにつれて減らすことができます。

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。1 ページに保存されるローの最大数は 255 であるため、小さいローを持つテーブルでは、各ページは一杯になりません。たとえば、8 KB のページ全体を使用するには、ローの平均サイズを最低 32 バイトにしてください。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。

クワイエット・モードで処理を実行する (-q) 出力メッセージを表示しません。

チェックサムを使用する (-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っている場合は、ページがディスクで変更されたか破損しています。

トランザクション・ログ・ファイル名を設定する (-t) 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」491 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。ファイル名にパスがない場合、そのファイルはデータベース・ファイルと同じディレクトリに置かれます。−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。

照合順 (-z) データベース中のすべての文字列比較で使用される照合順です。

-z を指定しないと、Adaptive Server Anywhere によってオペレーティング・システムの言語と文字セットの設定に基づいたデフォルト照合が選択されます。

詳細については、「照合の選択」438 ページを参照してください。

既存のデータベースが使用している照合を変更するには、データベースをアンロードし、適切な照合を使用して新しいデータベースを作成してから、データベースを再ロードする必要があります。データの変換も必要になることがあります。

カスタム照合を作成する場合は、照合ユーティリティを使用して照合を含むファイルを作成します。照合を修正して、適切なスクリプトに挿入したら、初期化ユーティリティを使用してデータベースを作成し、新しい照合を指定します。

カスタム照合ファイルで照合ラベルを変更してください。変更しないと、既存の照合ラベルと重複してしまうため、初期化ユーティリティを使って新しい照合を挿入できません。

カスタム照合順の詳細については、「国際言語と文字セット」417 ページを参照してください。

照合ユーティリティの詳細については、「照合ユーティリティ」653 ページを参照してください。

Interactive SQL ユーティリティ

Interactive SQL は、データベースをブラウズし、SQL 文をデータベース・サーバへ送信するための対話型環境をユーザに提供します。

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] ー [SQL Anywhere 9] ー [Adaptive Server Anywhere] ー [Interactive SQL] の順に選択する。
- コマンド・プロンプトで、dbisql コマンドを入力する。

Interactive SQL のみから使用される SQL 文

以下は Interactive SQL のみから使用できる SQL 文のリストです。

『ASA SQL リファレンス・マニュアル』> 「CLEAR 文 [Interactive SQL]」

『ASA SQL リファレンス・マニュアル』>「CONFIGURE 文 [Interactive SQL]」

『ASA SQL リファレンス・マニュアル』> 「CONNECT 文 [ESQL] [Interactive SQL]」

『ASA SQL リファレンス・マニュアル』> 「DISCONNECT 文 [ESQL] [Interactive SQL]」

『ASA SQL リファレンス・マニュアル』> 「EXIT 文 [Interactive SQL]」

『ASA SQL リファレンス・マニュアル』> 「HELP 文 [Interactive SQL]」

『ASA SQL リファレンス・マニュアル』> 「INPUT 文 [Interactive SQL]」

『ASA SQL リファレンス・マニュアル』> 「OUTPUT 文 [Interactive SQL]」

『ASA SQL リファレンス・マニュアル』> 「PARAMETERS 文 [Interactive SQL]」

『ASA SQL リファレンス・マニュアル』>「READ 文 [Interactive SQL]」

『ASA SQL リファレンス・マニュアル』>「SET CONNECTION 文 [Interactive SQL] [ESQL]」

『ASA SQL リファレンス・マニュアル』>「SET OPTION 文 [Interactive SQL]」

『ASA SQL リファレンス・マニュアル』>「START DATABASE 文 [Interactive SQL]」

『ASA SQL リファレンス・マニュアル』>「START ENGINE 文 [Interactive SQL]」

『ASA SQL リファレンス・マニュアル』>「START LOGGING 文 [Interactive SQL]」

『ASA SQL リファレンス・マニュアル』>「STOP LOGGING 文 [Interactive SQL]」

『ASA SQL リファレンス・マニュアル』>「SYSTEM 文 [Interactive SQL]」

Sybase Central から Interactive SQL を起動する

❖ Sybase Central から Interactive SQL を開くには、次の手順に従います。

1. 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。
2. 右ウィンドウ枠にある [ユーティリティ] タブをクリックします。
3. 右ウィンドウ枠の [Interactive SQL を開く] をダブルクリックします。

   Interactive SQL ウィンドウが表示されます。

ヒント
Interactive SQL へは以下の手順で Sybase Central からもアクセスできます。

- [ツール] ー [Adaptive Server Anywhere 9] ー [Interactive SQL を開く] を選択する。
- 左ウィンドウ枠でデータベースを選択し、[ファイル] ー [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。

dbisql コマンド・ライン・ユーティリティを使用して Interactive SQL を開く

構文 **dbisql** [*options*] [*dbisql-command* | *command-file*]

オプション	説明
-c *"keyword=value; …"*	データベース接続パラメータを指定する
-codepage *codepage*	ファイルの読み込みまたは書き込みをするときに使用するコード・ページを指定する
-d *delimiter*	指定の文字列をコマンド・デリミタとして使用する
-d1	文が実行されるときにその文を出力する (コマンド・プロンプト・モードのみ)

オプション	説明
-datasource *dsn-name*	接続先の ODBC データ・ソースを指定する
-f *filename*	*filename* というファイルを (実行せずに) 開く
-host *hostname*	データベース・サーバを実行中のマシンのホスト名か IP アドレスを指定する
-jConnect	jConnect を使用してデータベースに接続する
-nogui	コマンド・プロンプト・モードで実行する
-ODBC	iAnywhere JDBC ドライバを使用してデータベースに接続する
-onerror { **continue** | **exit** }	すべてのユーザに対して ON_ERROR オプションを上書きする
-port *portnumber*	データベース・サーバの指定のポート番号を調べる
-q	クワイエット・モード - ウィンドウまたはメッセージは表示しない (エラー・メッセージは表示されることに注意)
-x	構文チェックのみ (コマンドは実行しない)

説明

Interactive SQL を使うと、SQL コマンドの入力、または Interactive SQL コマンド・ファイルの実行ができます。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。

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。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

Interactive SQL では、QUOTED_IDENTIFIER データベース・オプションを ON に設定する必要があります。この設定によって、一部の文や多くのデータベース関数が正常に機能するためです。Interactive SQL はデータベースに接続すると、このオプションを自動的に ON に設定します。

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*。

Interactive SQL ユーティリティのオプション

接続パラメータ (-c) 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」236 ページを参照してください。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。

コード・ページ (-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 NT マシンでは Interactive SQL 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
```

サポートされるコード・ページのリストについては、「サポートされているコード・ページ」473 ページを参照してください。

コマンド・デリミタ (-d) 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。

コマンド・ラインで指定したコマンド・デリミタは、データベースに保存されている設定 (ユーザ用または PUBLIC 設定) に関係なく、その Interactive SQL セッション中のすべての接続で使用されます。

エコー文 (-d1)　(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)。Interactive SQL は、実行する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。

データ・ソース (-datasource)　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。

Interactive SQL を使用してファイルを開く (-f)　*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、ファイル名にブランク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。

ホスト (-host)　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。現在のマシンを意味する **localhost** を使用できます。

jConnect を使用する (-jConnect)　Sybase jConnect JDBC ドライバを使用してデータベースに接続します。

コマンド・プロンプト・モードで実行する (-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。終了コードには次のものがあります。

プログラム終了コード	説明
0	成功
1	一般的な失敗。ある時点で SQL か Interactive SQL 文が失敗し、SQL 文の実行中止をユーザが選択した。または、Interactive SQL が内部エラーを通知した。
5	ユーザが Interactive SQL を終了した。実行中にエラーが発生すると、エラーを無視するか、Interactive SQL を停止するか終了するかが要求される。終了を選択した場合は、プログラムはコード 5 を返す。コード 5 は、Interactive SQL オプション ON_ERROR に EXIT が設定されているときにエラーが発生した場合にも返される。
9	接続できない
255	不正なコマンド。コマンドに、不完全なまたは無効なオプションが含まれている。

iAnywhere JDBC ドライバを使用する (-ODBC) iAnywhere JDBC ドライバを使用して接続します。これはデフォルトの方法であり、ほとんどの状況でおすすめします。

ON_ERROR オプション設定を上書きする (-onerror) 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。

詳細については、「ON_ERROR オプション [Interactive SQL]」874 ページ を参照してください。

データベース・サーバ・ポート (-port) 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。Adaptive Server Anywhere のデフォルト・ポート番号は **2638** です。

クワイエット・モードで処理を実行する (-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ても、エラー・メッセージは表示されることに注意してください。

構文チェックのみ (-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、『ASA SQL リファレンス・マニュアル』> 「SQL 言語の要素」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 by passwd
```

言語ユーティリティ

言語ユーティリティは、Adaptive Server 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。

dblang コマンド・ライン・ユーティリティを使用した言語の管理

構文 **dblang** [*options*] [*language-code*]

オプション	説明
-q	クワイエット・モードで作動する (メッセージを表示しない)

言語コード	言語
DE	ドイツ語
EN	英語
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

言語コード	言語
ZH	中国語 (簡体文字)

説明

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。

- **Adaptive Server 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、Adaptive Server 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、*dblgXX9.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、自分のマシン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 Studio 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。

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。

言語ユーティリティのオプション

クワイエット・モードで処理を実行する (-q)　出力メッセージを表示しません。

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
```

ライセンス・ユーティリティ

ライセンス・ユーティリティは、ネットワーク・データベース・サーバに対してライセンス・ユーザを追加します。

dblic コマンド・ライン・ユーティリティを使用したライセンスの管理

構文

dblic [*options*] *executable-name user-name company-name*

オプション	説明
@data	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出す
-l *type*	ライセンス・タイプを指定する。指定可能な値は **perseat** または **processor**。
-o *filename*	ファイルに出力メッセージのログを取る
-p *operating-system*	オペレーティング・システムを指定する。指定可能な値は **WIN32**、**WIN64**、**NetWare**、および **UNIX**。
-q	クワイエット・モードで作動する (メッセージを表示しない)
-u *license-number*	ユーザまたはプロセッサ・ライセンスの総数を指定する
executable-name	ライセンスされている実行プログラムを指定する。パスは現在のディレクトリの相対パス。Adaptive Server Anywhere ネットワーク・データベース・サーバまたは Mobile Link 同期サーバのいずれかになる。
user-name	ライセンスのユーザ名を指定する
company-name	ライセンスの会社名を指定する

説明

ライセンス・ユーティリティは、ネットワーク・データベース・サーバに対してライセンス・ユーザを追加します。このユーティリティは、Sybase 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は、サーバの現在のライセンス情報をサーバを起動せずに表示するためにも使用できます。

dblic コマンドを実行するたびに、ライセンス情報が更新されます。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

UNIX では、dbsrv9 と dbmlsrv9 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、ライセンス [dblic] ユーティリティをこれらの実行プログラムと共に使用すると失敗します。dbsrv9 と dbmlsrv9 実行プログラムは、ライセンス・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。

ライセンス・ユーティリティは、NetWare ではサポートされていません。NetWare で Adaptive Server Anywhere 実行プログラムをライセンスするには、NetWare 上の Adaptive Server 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、Adaptive Server 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 マシンからライセンス・ユーティリティを実行してください。

ライセンス・ユーティリティのオプション

@data 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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

ライセンス・タイプ (-l) 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。有効なエントリは **perseat** と **processor** です。

ファイルに出力メッセージのログを取る (-o) 指定したファイルに、出力メッセージを書き込みます。

オペレーティング・システム (-p)　ライセンスされたオペレーティング・システムを入力します。指定可能な値は次のとおりです。

- **NetWare**　Novell NetWare オペレーティング・システム
- **UNIX**　UNIX オペレーティング・システム
- **WIN32**　Windows 95/98/Me、Windows NT/2000/XP、または Windows CE オペレーティング・システム
- **WIN64**　64 ビット Windows オペレーティング・システム

クワイエット・モードで処理を実行する (-q)　出力メッセージを表示しません。

ライセンス番号 (-u)　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。

実行プログラム名　ライセンスされているネットワーク・データベース・サーバの実行プログラム (*dbsrv9.exe*) または Mobile Link 同期サーバ (*dbmlsrv9.exe*) のパスとファイル名を入力します。パスは、絶対パスか、現在の作業ディレクトリに対する相対パスのいずれかにします。

サーバ実行プログラムの名前だけを入力することで、サーバを起動せずに、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。

ユーザ名　ライセンスのユーザ名。この名前は、起動時にデータベース・サーバの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。

会社名　ライセンスの会社名。この名前は、起動時にデータベース・サーバの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。

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 NT 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 -p WIN32 -u 50 dbsrv9.exe "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。

```
Licensed nodes: 50
 User: Sys Admin
 Company: My Co
```

Log Transfer Manager

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 Adaptive Server Anywhere データベースに必要です。

dbltm コマンド・ライン・ユーティリティの使用

構文　**dbltm** [*options*]

パラメータ

オプション	説明
@data	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出す
-A	アップデートをフィルタしない
-C *config_file*	指定の設定ファイルを使用する
-I *interface_file*	指定の interfaces ファイルを使用する
-M	リカバリ・モード
-S *LTM_name*	LTM 名を指定する
-dl	画面にログ・メッセージを表示する
-ek *key*	暗号化キーを指定する
-ep	暗号化キーを入力するよう要求する
-o *filename*	ファイルに出力メッセージのログを取る
-os *size*	出力ファイルの最大サイズ
-ot *file*	ファイルをトランケートして、ファイルに出力メッセージを記録する
-q	最小化ウィンドウで実行する

オプション	説明
-s	ログ転送言語 (LTL) コマンドを表示する
-ud	デーモンとして実行する (UNIX)
-v	冗長モード

説明

Adaptive Server 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。

Log Transfer Manager ユーティリティのオプション

@data 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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

アップデートをフィルタしない (-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。

設定ファイルを使用する (-C)　設定ファイル *config_file* を使用して、LTM 設定を決定します。デフォルト設定ファイルは *dbltm.cfg* です。

設定ファイルについては、「LTM 設定ファイル」717 ページを参照してください。

ログ転送言語メッセージを表示する (-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。

暗号化キーを指定する (-ek)　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。強力に暗号化されたデータベースの場合、-ek または -ep をどちらか指定します。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。

暗号化キーを入力するよう要求する (-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。強力に暗号化されたデータベースの場合、-ek または -ep をどちらか指定します。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。

interfaces ファイルを使用する (-I)　(i の大文字) 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 SQLEDIT で作成したファイルです。デフォルトの interfaces ファイルは、*Sybase* ディレクトリの *ini* サブディレクトリにある *SQL.ini* です。

リカバリ・モード (-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。

ファイルに出力メッセージのログを取る (-o) 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。

出力ファイルのサイズを制限する (-os) 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。

ログ・ファイルをトランケートし、指定したログ・ファイルを使用する (-ot) 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。

クワイエット・モードで処理を実行する (-q) LTM の起動時にウィンドウを最小化します。

LTM 名を指定する (-S) 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、SQLEDIT に入力した LTM に対する Open Server 名と対応させます。

すべての LTL コマンドを記録する (-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。

デーモンとして実行する (-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。

冗長モードで動作する (-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。

LTM 設定ファイル

Adaptive Server 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、Adaptive Server 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、Adaptive Server Anywhere LTM では使用されないパラメータには、「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、Adaptive Server Anywhere LTM 設定ファイルのみにある。
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。このエントリは、Adaptive Server Anywhere LTM 設定ファイルのみにある。
backup_only	デフォルトは **off**。**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する。
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。バッチ・モードは全体的なスループットを向上させるが、応答時間が長くなることがある。
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。

パラメータ	説明
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。
Continuous	デフォルトはオン。オフに設定すると、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する。
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。
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。レプリケーション・システム内のすべてのソート順を同じにする。 デフォルトのソート順はバイナリ・ソート。
maint_cmds_to_skip	(無視される)
qualify_table_owner	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 を送信する。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る。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。デフォルトは **off**。
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。デフォルトは **off**。

パラメータ	説明
Retry	失敗した Adaptive Server 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。
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る。ほとんどの設定は、RS_Source_db と SQL_database 設定オプションの両方で同じ設定。
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る。ほとんどの場合、RS_Source_ds と SQL_server 設定オプションはいずれも同じ設定内容を使用する。
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。
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る。Adaptive Server 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ない。このため、Adaptive Server Enterprise LTM の *scan_retry* 値よりも小さい数にできる。
skip_ltl_cmd_err	(無視される)
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。Adaptive Server 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・ログを見つける。

パラメータ	説明
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。このエントリは、Adaptive Server Anywhere LTM 設定ファイルのみにある。
SQL_pw	SQL_user ユーザ ID に対するパスワード
SQL_server	LTM が接続するプライマリ・サイト Adaptive Server 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。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・ログを見つける。
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

ログ変換ユーティリティを使って、トランザクション・ログを SQL コマンド・ファイルに変換できます。

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。

[ログ・ファイル変換] ウィザードを使用したトランザクション・ログの変換

❖ **トランザクション・ログをコマンド・ファイルに変換するには、次の手順に従います。**

1. 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。
2. 右ウィンドウ枠にある [ユーティリティ] タブをクリックします。
3. 右ウィンドウ枠で、[ログ・ファイルの変換] をダブルクリックします。

   [ログ・ファイル変換] 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

Sybase Central では、次の方法で [ログ・ファイル変換] ウィザードを利用することもできます。

- [ツール] ー [Adaptive Server Anywhere 9] ー [ログファイルの変換] を選択する。

- 左ウィンドウ枠でデータベースを選択し、[ファイル] ー [ログ・ファイルの変換] 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 [ログ・ファイルの変換] を選択する。

dbtran コマンド・ライン・ユーティリティを使用したトランザクション・ログの変換

構文

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*]

データベース・サーバに対して実行する場合。

dbtran [*options*]

オプション	説明
@data	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出す
-a	コミットされていないトランザクションを含める
-c *"keyword=value; …"*	データベース接続パラメータを指定する (トランザクション・ログ名と一緒には使用できない)
-d	出力を日付順に表示する
-ek *key*	暗号化キーを指定する
-ep	暗号化キーを入力するよう要求する
-f	最終チェックポイント以降だけを出力する
-g	監査レコードを出力に含める
-ir *offset1,offset2*	指定された 2 つのオフセット間のログの部分だけを含める

オプション	説明
-is *source,…*	指定されたソースで作成されたローだけを含める
-it *user.table,…*	カンマで区切られたユーザ・テーブルのリストを指定して、そこに指定されているテーブルの操作だけを含める
-j *date/time*	任意の日時より前の最終チェックポイントから出力する
-m	トランザクション・ログのディレクトリを指定する (-n オプションの指定が必要)
-n *filename*	データベース・サーバに対して使用すると、SQL ファイルを出力する
-o *filename*	ファイルに出力メッセージのログを取る
-q	クワイエット・モードで動作する (メッセージを表示しない)
-r	コミットされていないトランザクションを削除する (デフォルト)
-rsu *username,…*	デフォルトの Replication Server ユーザ名を上書きする
-s	ANSI 標準の SQL UPDATE トランザクションを生成する
-sr	SQL Remote コメントを生成する
-t	出力にトリガ生成トランザクションを含める
-u *userid,…*	リストされたユーザのトランザクションだけを変換する
-x *userid,…*	リストされたユーザのトランザクションを排除する
-y	確認メッセージを表示せずにファイルを置き換える
-z	トリガ生成トランザクションをコメントとしてのみ含める

オプション	説明
transaction-log	変換するログ・ファイル
SQL-file	変換した情報を含む出力ファイル

説明

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 **asademo9** からログ情報を変換して、出力をファイル *asademo.SQL* に入れるものです。

  ```
  dbtran -c
  "eng=asademo9;dbn=asademo;uid=DBA;pwd=SQL" -n
  asademo.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 asademo.log asademo.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 asademo.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。

複数のログ・ファイルにまたがるトランザクションのリカバリについては、「複数のトランザクション・ログからのリカバリ」547 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。

ログ変換ユーティリティのオプション

@data　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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

コミットされていないトランザクションを含む (-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。

接続文字列 (-c) 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。

dbtran の実行には DBA 権限が必要です。

接続パラメータの詳細については、「接続パラメータ」236 ページを参照してください。

日付順に出力する (-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。

暗号化キーを指定する (-ek) 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。

強力に暗号化されたデータベースの場合、-ek または -ep をどちらか指定します。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。

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
```

暗号化キーを入力するよう要求する (-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。

強力に暗号化されたデータベースの場合、-ek または -ep をどちらか指定します。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。

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
```

最終チェックポイント以降だけ出力する (-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。

監査情報を含める (-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。

詳細については、「AUDITING オプション [データベース]」 824 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。

オフセットの範囲を含む (-ir)　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。

指定したソースで作成されたローだけを含む (-is)　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。

指定のテーブルを含む (-it) 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。

任意の日付より前の最終チェックポイントから出力する (-j) 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。

トランザクション・ログ・ディレクトリ (-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。

出力ファイル (-n) 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。

ファイルに出力メッセージのログを取る (-o) 指定したファイルに、出力メッセージを書き込みます。

クワイエット・モードで処理を実行する (-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。このオプションを指定する場合、-y オプションも指定しないと操作は失敗します。

コミットされていないトランザクションを含まない (-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。

Replication Server ユーザ名を無効にする (-rsu) 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。

ANSI 標準の SQL UPDATE を生成する (-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。

SQL Remote コメントを生成する (-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。

トリガによって生成されたトランザクションを含む (-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。

リストされたユーザのトランザクションだけを出力する (-u)　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。

リストされたユーザ以外のトランザクションを出力する (-x)　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。

確認メッセージを表示することなく処理を実行する (-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。

トリガ生成トランザクションをコメントとしてのみ含める (-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。

Ping ユーティリティ

Ping ユーティリティは、接続に関する問題のトラブルシューティングを支援します。

dbping コマンド・ライン・ユーティリティを使用した接続のトラブルシューティング

構文 **dbping** [*options*]

オプション	説明
@data	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出す
-c *"keyword=value; …"*	データベース接続パラメータ
-d	サーバが検出されたらデータベース接続を作成する
-l *library*	指定した ODBC ドライバまたはライブラリをロードする
-m	ODBC ドライバ・マネージャを使用する
-o *filename*	ファイルに出力メッセージのログを取る
-pc *property,…*	指定した接続プロパティについてレポートする
-pd *property,…*	指定したデータベース・プロパティについてレポートする
-ps *property,…*	指定したデータベース・サーバ・プロパティについてレポートする
-q	クワイエット・モードで作動する (メッセージを表示しない)
-z	デバッグ情報を表示する

説明

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。

Ping ユーティリティのオプション

@data 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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

接続パラメータ (-c) 接続パラメータの詳細については、「接続パラメータ」236 ページを参照してください。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。

データベース接続を確立する (-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。-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にのみ、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=asademo"
```

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=asademo" -d
```

指定のライブラリをロードする (-l) 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=ASA 9.0 Sample" -l dbodbc9
```

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。

ODBC を使用して接続する (-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。

ファイルに出力メッセージのログを取る (-o) 指定したファイルに、出力メッセージを書き込みます。

接続のプロパティをレポートする (-pc) 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。

接続プロパティのリストについては、「接続レベルのプロパティ」923 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
```

データベースのプロパティをレポートする (-pd) 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・プロパティのリストについては、「データベース・レベルのプロパティ」946 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 Java のバージョンを表示します。

```
dbping -c ... -pd JDKVersion
```

データベース・サーバのプロパティをレポートする (-ps)　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。

データベース・サーバ・プロパティのリストについては、「サーバ・レベルのプロパティ」936 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
```

クワイエット・モードで処理を実行する (-q)　dbping が失敗すると、メッセージは必ず表示されます。dbping が成功した場合、-q が指定されているときにはメッセージは表示されません。

デバッグ情報を表示する (-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。

再構築ユーティリティ

Rebuild バッチ・ファイル、コマンド・ファイル、またはシェル・スクリプトなど、データベースを再構築する一連のユーティリティを呼び出すファイルを使用してデータベースを再構築できます。Sybase Central で [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] ウィザードの詳細については、「アンロード・ユーティリティ」762 ページを参照してください。

REBUILD バッチまたはコマンド・ファイルを使用したデータベースの再構築

構文

rebuild *old-database new-database* [*DBA-password*]

参照

- 「dbunload コマンド・ライン・ユーティリティを使用したデータベースのアンロード」765 ページ
- 「dbinit コマンド・ライン・ユーティリティを使用したデータベースの作成」688 ページ
- 「Interactive SQL ユーティリティ」698 ページ

説明

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・ファイルを使わずに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。

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。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

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。*

サーバ検索ユーティリティ

サーバ検索ユーティリティは、直接 TCP/IP ネットワーク上のデータベース・サーバを検索して、接続トラブルの診断を支援します。

dblocate コマンド・ライン・ユーティリティを使用したサーバの検索

構文

dblocate [*options*] [*server-name*]

オプション	説明
@data	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出す
-n	IP アドレスをマシン名に解決しない
-o *filename*	ファイルに出力メッセージのログを取る
-q	クワイエット・モードで作動する (メッセージを表示しない)
server-name	リストを指定したマシンで実行するサーバに制限するホスト名または IP アドレス

説明

dblocate ユーティリティは、直接接続されているネットワーク上の TCP/IP で実行されている Adaptive Server Anywhere データベース・サーバを検索します。データベース・サーバとそのアドレスのリストを表示します。

ネットワークによっては、ユーティリティが結果を表示するのに数秒かかる場合もあります。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

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、クライアントとサーバ検索ユーティリティ (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 と共に使用し、ネットワーク・サーバ上でのみ使用されます。

LDAP の詳細については、「LDAP サーバを使用した接続」119 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、サーバ検索ユーティリティは -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 アドレスを持つマシン (たとえば、マシン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・マシンで実行し、同じサーバ名を持つパーソナル・サーバをローカル・マシンで実行している場合です。

サーバ検索ユーティリティのオプション

@data　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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

IP アドレスをマシン名に解決しない (-n)　マシン名ではなく IP アドレスを出力にリストします。マシン名の検索は遅いため、これによりパフォーマンスが向上する場合があります。

ファイルに出力メッセージのログを取る (-o)　指定したファイルに、出力メッセージを書き込みます。

クワイエット・モードで処理を実行する (-q)　出力メッセージを表示しません。

server-name　指定した IP アドレスまたはホスト名を持つマシンで実行しているデータベース・サーバのみをリストします。–n オプションを指定する場合、*server-name* 値は IP アドレスにしてください。–n を指定しない場合、*server-name* にはホスト名を指定します。たとえば、次のコマンドは、マシン **jfrancis** 上のサーバを検索します。

```
dblocate jfrancis
```

サービス作成ユーティリティ

サービス作成ユーティリティは、Adaptive Server Anywhere サービスを作成、削除、変更するために使用するツールです。次の方法で、サービス作成ユーティリティにアクセスできます。

- Sybase Central の [サービス作成] ウィザードを使用する。
- コマンド・プロンプトで、dbsvc コマンドを入力する。

[サービス作成] ウィザードを使用したサービスの作成

❖ **サービスを作成するには、次の手順に従います。**

1. Sybase Central の左ウィンドウ枠で、Adaptive Server Anywhere 9 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] ー [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。

dbsvc コマンド・ライン・ユーティリティを使用したサービスの管理

構文

dbsvc [**-q**] [**-y**] **-d** <*svc*>

dbsvc [**-q**] **-g** <*svc*>

dbsvc [**-q**] **-l**

dbsvc [**-q**] [**-y**] <*creation options*> **-w** <*svc*> <*details*>

dbsvc [**-q**] **-u** <*svc*>

dbsvc [**-q**] **-x** <*svc*>

details:

<*full-executable-path*> [*options*]

主要オプション	説明
@data	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出す
-d *service_name*	サービスを削除する
-g *service_name*	サービスの詳細を取得する
-l	すべての Adaptive Server Anywhere サービスをリストする
-u *service_name*	*service_name* という名前のサービスを起動する
-w *executable parameters*	サービスを作成する
-x *service_name*	*service_name* という名前のサービスを停止する

作成オプション	説明
-a *acct*	使用するアカウント名 (-p と一緒に使用)
-as	ローカル・システム・アカウントを使用する
-i	デスクトップとの対話をサービスに許可する
-p	アカウントのパスワード (-a と一緒に使用)
-rg *dependency*,…	サービスを作成するときのグループ依存性を指定する
-rs *dependency*,…	サービスを作成するときのサービス依存性を指定する
-s *startup*	起動オプション (デフォルトは Manual)。Automatic、Manual、または Disabled を指定する。
-sd *description*	サービスの説明を表示する
-sn *name*	サービスの名前を指定する

作成オプション	説明
-t *type*	サービスのタイプを指定する

変更オプション	説明
-cm	サービスの作成コマンドを表示する
-q	バナーを表示しない
-y	確認メッセージを表示せずにサービスを削除または上書きする

説明

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、Windows 上で動作している Adaptive Server Anywhere サービスを包括的な方法で管理できます。サービス作成ユーティリティは、Sybase Central での [サービス作成] ウィザードと同じ機能を提供します。

サービス作成ユーティリティを使用するには、ローカル・マシン上で Administrators グループのメンバである必要があります。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

サービスの詳細については、「Windows サービスの概要」30 ページを参照してください。

サービス作成ユーティリティのオプション

サービス作成ユーティリティを使用するときは、主要オプション、変更子オプション、詳細オプションを選択できます。

主要オプション

@data 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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

サービスを削除する (-d)　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。

サービスの詳細を取得する (-g)　パスワード以外のサービスの定義をリストします。

すべての Adaptive Server Anywhere サービスをリストする (-l)　使用できる Adaptive Server Anywhere サービスをリストします。

サービスを起動する (-u)　*service_name* という名前のサービスを起動します。

サービスを作成する (-w)　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 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。詳細については、次を参照してください。

- **dbsrv9 と dbeng9**　「データベース・サーバ」154 ページ
- **dbmlsrv9**　『Mobile Link 管理ガイド』>「Mobile Link 同期サーバのオプション」
- **dbmlsync**　『Mobile Link クライアント』>「Mobile Link 同期クライアント」
- **dbremote**　『SQL Remote ユーザーズ・ガイド』>「Message Agent」

サービスを停止する (-x) *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

アカウント名 (-a) 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。

ローカル・システム・アカウントを使用する (-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 オプションを使用すると、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。

デスクトップとの対話をサービスに許可する (-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・ウィンドウが表示されます。

アカウントのパスワード (-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、-a オプションと一緒にこのオプションを使用します。

グループ依存性を設定する (-rg) リストに含まれる各グループから少なくとも 1 つのサービスが起動されてから、作成したサービスの起動を許可します。

サービス依存性を設定する (-rs) 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。

起動オプション (-s) Adaptive Server 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。

サービスの説明 (-sd) 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。

サービス名 (-sn) 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

Adaptive Server Anywhere - *<svc>* です。たとえば、次のサービスにはデフォルトで Adaptive Server Anywhere - myserv という名前が付けられます。

```
dbsvc -as -w myserv
 "C:¥Program Files¥Sybase¥SQL Anywhere
9¥win32¥dbeng9.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 (すべて 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¥Sybase¥SQL Anywhere
9¥win32¥dbeng9.exe"
```

サービスのタイプ (-t)　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。

タイプ	説明
Network	Adaptive Server Anywhere ネットワーク・データベース・サーバ (dbsrv9)
Standalone	Adaptive Server Anywhere パーソナル・データベース・サーバ (dbeng9)
DBRemote	SQL Remote Message Agent (dbremote)
Mobile Link	Mobile Link 同期サーバ (dbmlsrv9)
DBMLSync	Mobile Link 同期クライアント (dbmlsync)

デフォルト設定は **Standalone** です。

変更子オプション

サービスの作成コマンドを表示する (-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。作成コマンドは別のマシン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、マシンに 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。

バナーを表示しない (-q) 情報バナーを表示しません。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。

確認メッセージを表示せずにサービスを削除または上書きする (-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。

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。サーバは、**LocalSystem** ユーザとして実行されます。

```
dbsvc -as -w myserv "C:¥Program Files¥Sybase¥SQL
Anywhere 9¥win32¥dbeng9.exe" -n myeng -c 8m "C:¥Program
Files¥Sybase¥SQL Anywhere 9¥sample.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。サーバはローカル・アカウントで実行され、マシン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¥Sybase¥SQL Anywhere
9¥win32¥dbsrv9.exe" -x tcpip -c 8m "C:¥Program
Files¥Sybase¥SQL Anywhere 9¥sample.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¥Sybase¥SQL Anywhere
9¥win32¥dbmlsync.exe" -c "dsn=ultralite 9.0 sample"
```

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¥Sybase¥SQL Anywhere
9¥win32¥dbeng9.exe"
```

プロセス生成ユーティリティ

バックグラウンドでデータベース・サーバを起動するユーティリティです。

dbspawn コマンド・ライン・ユーティリティを使用したバックグラウンドでのサーバの実行

構文

dbspawn [*options*] *server-command*

オプション	説明
@data	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出す
-f	実行中のサーバをチェックしない
-p	オペレーティング・システムのプロセス ID をレポートする
-q	クワイエット・モード (メッセージを表示しない)
server-command	データベース・サーバを開始するためのコマンド・ラインを指定する

説明

dbspawn ユーティリティは、バックグラウンドでサーバを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。*server-command* に指定したサーバがすでに実行されている場合、dbspawn は失敗をレポートします。

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。

指定したパスに少なくとも 1 つのスペースが含まれており、バッチ・ファイルからプロセス生成ユーティリティを実行している場合、1 組の二重引用符でそのパスを囲みます。次に例を示します。

```
dbspawn dbeng9 "C:¥Program Files¥Sybase¥SQL Anywhere
9¥asademo.db"
```

指定したパスに少なくとも 1 つのスペースが含まれており、コマンド・プロンプトからプロセス生成ユーティリティを実行している場合、追加の引用符セットを提供し、データベース・ファイルの前後でその引用符をエスケープしてください。次に例を示します。

```
dbspawn dbeng9 "¥"C:¥Program Files¥Sybase¥SQL Anywhere
9¥asademo.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。

プロセス生成ユーティリティのオプション

@data　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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

実行中のサーバをチェックしない (-f)　データベース・サーバがすでに実行されている場合、dbspawn コマンドがそのサーバを使用することがあります。このオプションを指定すると、dbspawn が実行されるたびに強制的に新しいデータベース・サーバが起動されます。

プロセス ID をレポートする (-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 dbeng9 -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
```

クワイエット・モードで処理を実行する (-q)　出力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。

サーバ・コマンドの詳細については、「データベース・サーバ」154 ページを参照してください。

停止ユーティリティ

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。

停止ユーティリティを実行できるのは、コマンド・プロンプトのみです。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。

停止 (dbstop) ユーティリティは NetWare では使用できません。

dbstop コマンド・ライン・ユーティリティを使用したデータベース・サーバの停止

構文

dbstop [*options*] [*server-name*]

オプション	説明
@data	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出す
-c *"keyword=value; ..."*	接続パラメータ
-d	指定したデータベースだけを停止する
-o *filename*	ファイルに出力メッセージのログを取る
-q	クワイエット・モード (メッセージを表示しない)
-x	アクティブな接続がある場合は停止しない
-y	アクティブな接続がある場合でもプロンプトを表示せずに停止する
server-name	停止するローカル・データベース・サーバの名前

説明

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。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

停止 [dbstop] ユーティリティは NetWare では使用できません。

停止ユーティリティのオプション

@data 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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

接続パラメータ (-c) 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。

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。**unconditional=YES** を指定すると、サーバ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ないで停止します。

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。

詳細については、「接続パラメータ」236 ページ、「Unconditional 接続パラメータ [UNC]」274 ページ、および 「-gk サーバ・オプション」192 ページを参照してください。

データベースのみを停止する (-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。

ファイルに出力メッセージのログを取る (-o)　指定したファイルに、出力メッセージを書き込みます。

クワイエット・モードで処理を実行する (-q)　データベースが実行されていなくても、メッセージを表示しません。

アクティブな接続がある場合は停止しない (-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。

プロンプトを表示せずに停止する (-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。

server-name　現在のマシン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。

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。

詳細については、「-gk サーバ・オプション」192 ページを参照してください。

例

データベースを含まない **myserver** という名前のサーバを実行しています。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、データベース **asademo.db** が起動しています。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"
```

トランザクション・ログ・ユーティリティ

トランザクション・ログ・ユーティリティを使って、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)。

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。
- InteractiveSQL から ALTER DATABASE *dbfile* MODIFY LOG 文を使用する。

  詳細については、『ASA SQL リファレンス・マニュアル』>「ALTER DATABASE 文」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。

[ログ・ファイル設定の変更] ウィザードを使用したログ・ファイルの管理

❖ **トランザクション・ログ・ファイル名を変更するには、次の手順に従います。**

1. 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。

2　右ウィンドウ枠にある [ユーティリティ] タブをクリックします。

3　右ウィンドウ枠で、[ログ・ファイル設定の変更] をダブルクリックします。

[ログ・ファイル設定の変更] 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
[ログ・ファイル設定の変更] ウィザードには、[ツール] － [Adaptive Server Anywhere 9] － [ログ・ファイル設定の変更] を選択してアクセスすることもできます。

dblog コマンド・ライン・ユーティリティを使用したログ・ファイルの管理

構文　**dblog** [*options*] *database-file*

オプション	説明
@data	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出す
-ek *key*	暗号化キーを指定する
-ep	暗号化キーを入力するよう要求する
-g *n*	LTM の世代番号を *n* に設定する
-il	データベースに格納されている LTM トランケーション・オフセットを無視する
-ir	データベースに格納されている SQL Remote トランケーション・オフセットを無視する
-is	データベースに格納されている dbmlsync トランケーション・オフセットを無視する
-m *mirror-name*	トランザクション・ログ・ミラー名を設定する

オプション	説明
-n	トランザクション・ログやミラー・ログを使用しない
-o *file-name*	ファイルに出力メッセージのログを取る
-q	クワイエット・モード (メッセージを表示しない)
-r	トランザクション・ログ・ミラーを使用しない
-t *log-name*	トランザクション・ログ名を設定する
-x *n*	トランザクション・ログの現在の相対オフセットを *n* に設定する
-z *n*	トランザクション・ログの開始オフセットを *n* に設定する

説明

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。

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
- レプリケーションで使用する開始オフセット
- レプリケーションで使用する終了オフセット
- ページ・サイズ
- ページの総数
- 空きページの数
- 使用されているログ・ファイルの割合

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

トランザクション・ログ・ユーティリティのオプション

@data 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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

暗号化キーを指定する (-ek) 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。強力に暗号化されたデータベースの場合、-ek または -ep をどちらか指定します。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。

暗号化キーを入力するよう要求する (-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。強力に暗号化されたデータベースの場合、-ek または -ep をどちらか指定します。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。

世代番号を設定する (-g) 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。

LTM トランケーション・オフセットを無視する (-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。

SQL Remote トランケーション・オフセットを無視する (-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。

dbmlsync トランケーション・オフセットを無視する (-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。

トランザクション・ログ・ミラー・ファイルの名前を設定する (-m)

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。

トランザクション・ログを使用しない (-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限り、トランザクション・ログを削除することはできません。

ファイルに出力メッセージのログを取る (-o)　指定したファイルに、出力メッセージを書き込みます。

クワイエット・モードで処理を実行する (-q)　出力メッセージを表示しません。

トランザクション・ログ・ミラーを使用しない (-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。

トランザクション・ログ・ファイルの名前を設定する (-t)　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。

現在のログ・オフセットを設定する (-x)　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、現在のログ・オフセットをリセットし、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。

このオプションの使い方については、『SQL Remote ユーザーズ・ガイド』>「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」を参照してください。

開始ログ・オフセットを設定する (-z)　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、ログ・オフセットの開始をリセットし、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。

このオプションの使い方については、『SQL Remote ユーザーズ・ガイド』> 「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」を参照してください。

展開ユーティリティ (旧式)

展開ユーティリティを使うと、圧縮ユーティリティで作成した圧縮データベース・ファイルを展開できます。展開ユーティリティは、圧縮されたファイルを読み込み、データベース・ファイルを展開した状態にリストアします。

次の方法で、展開ユーティリティにアクセスできます。

- Sybase Central の [データベース展開] ウィザードを使用する。
- コマンド・プロンプトで、dbexpand 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。

推奨されなくなった機能
圧縮データベースは推奨されなくなりました。

[データベース展開] ウィザードを使用したデータベースの展開

❖ **圧縮データベース・ファイルを展開するには、次の手順に従います。**

1. 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。
2. 右ウィンドウ枠にある [ユーティリティ] タブをクリックします。
3. 右ウィンドウ枠にある [データベースの展開] をダブルクリックします。

   [データベース展開] 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
このウィザードには、[ツール] ー [Adaptive Server Anywhere 9] ー [データベースの展開] を選択してアクセスすることもできます。

dbexpand コマンド・ライン・ユーティリティを使用したデータベースの展開

構文

dbexpand [*options*] *compressed-database-file* [*database-file*]

オプション	説明
@data	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出す
-ek *key*	暗号化キーを指定する
-ep	暗号化キーを入力するよう要求する
-o *filename*	ファイルに出力メッセージのログを取る
-q	クワイエット・モードで作動する (メッセージを表示しない)
-y	確認メッセージを表示せずに既存の出力ファイルを消去する

説明

入力ファイル名の拡張子は、デフォルトで *cdb* です。出力ファイル名 (拡張子付) は、入力ファイル名 (拡張子付) と同じ名前にしないようにしてください。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

展開ユーティリティのオプション

@data 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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

暗号化キーを指定する (-ek)　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。強力に暗号化されたデータベースの場合、-ek または -ep をどちらか指定します。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。

暗号化キーを入力するよう要求する (-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。強力に暗号化されたデータベースの場合、-ek または -ep をどちらか指定します。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。

ファイルに出力メッセージのログを取る (-o)　指定したファイルに、出力メッセージを書き込みます。

クワイエット・モードで処理を実行する (-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。

確認メッセージを表示することなく処理を実行する (-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、既存のデータベース・ファイルが自動的に置き換えられます。

アンロード・ユーティリティ

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で区切った形で、指定ディレクトリのファイルの中へアンロードします。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、『ASA SQL ユーザーズ・ガイド』> 「[データのアンロード] ダイアログの使用」を参照してください。

アンロード・ユーティリティへのアクセス

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。
- 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 から実行してください。*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。さらに、*reload.sql* ファイルは、DBA ユーザ ID から実行してください (通常、これは、ユーザ ID が **DBA** (パスワード **SQL**) だけである新しいデータベースで実行されます)。

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。

詳細については、「-gl サーバ・オプション」193 ページを参照してください。

dbo によって所有されるオブジェクト

dbo ユーザ ID は、データベース内の Adaptive Server Enterprise 互換システム・オブジェクトを所有します。

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。データをアンロードするとき、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。

アンロードとレプリケーション

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。

詳細については、『SQL Remote ユーザーズ・ガイド』> 「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」と『SQL Anywhere Studio 新機能ガイド』> 「データベース・ファイル・フォーマットのアップグレード」を参照してください。

[データベース・アンロード] ウィザードを使用したデータベースのアンロード

[データベース・アンロード] ウィザードを使用すると、順を追って Sybase Cenetral 内のデータベースをアンロードすることができます。このウィザードを使用してデータベースをアンロードするときに、データベース内のすべてのテーブルをアンロードするのか、またはデータベースからテーブルのサブセットのみをアンロードするのかを選択できます。[所有者別にオブジェクトをフィルタ] ダイアログ内で選択したユーザ用のテーブルのみが、[データベース・アンロード] ウィザードに表示されます。特定のデータベース・ユーザに属するテーブルを表示させたい場合、アンロード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 [所有者別にオブジェクトをフィルタ] を選択して、表示されるダイアログから対象ユーザを選択します。

[データベース・アンロード] ウィザードでは、再ロード・ファイルではなく、既存のデータベースへ再ロードするオプションもあります。これを実行するには、Sybase Central 内のアンロードするデータベースと再ロードするデータベースの両方と接続する必要があります。

❖ データベース・ファイルまたは実行中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。

1. 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。
2. 右ウィンドウ枠にある [ユーティリティ] タブをクリックします。
3. 右ウィンドウ枠にある [データベースのアンロード] をダブルクリックします。

   [データベース・アンロード] 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

Sybase Central では、次の方法で [データベース・アンロード] ウィザードを利用することもできます。

- [ツール] ー [Adaptive Server Anywhere 9] ー [データベースのアンロード] を選択する。
- 左ウィンドウ枠でデータベースを選択し、[ファイル] ー [データベースのアンロード] 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 [データベースのアンロード] を選択する。

Sybase Central からデータベースをアンロードする方法については、『ASA SQL ユーザーズ・ガイド』> 「データベースのエクスポート」を参照してください。

dbunload コマンド・ライン・ユーティリティを使用したデータベースのアンロード

構文　**dbunload** [*options*] [*directory*]

オプション	説明
@data	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出す
-ac *"keyword=value; …"*	再ロードの接続パラメータを指定する
-an *database*	アンロードするデータベースと同じ設定でデータベース・ファイルを作成し、それを自動的に再ロードする
-ap *size*	新規データベースのページ・サイズを指定する (-an または -ar も指定する必要がある)
-ar [*directory*]	データベースを再構築して置き換える
-c *"keyword=value; …"*	アンロードのためのデータベース接続パラメータを指定する
-d	データのみをアンロードする
-e *table, …*	リストされたテーブルはアンロードしない
-ea *algorithm*	データベースの暗号化に使用する強力な暗号化アルゴリズムを指定する。AES か AES_FIPS を選択できる。
-ek *key*	新しいデータベースの暗号化キーを指定する
-ep	新しいデータベースの暗号化キーを指定する
-ii	内部アンロード、内部再ロード (デフォルト)
-ix	内部アンロード、外部再ロード

オプション	説明
-jr	アンロードまたは再ロード時に Java を無視する
-m	データベースのレプリケーションでユーザ ID を保存しない
-n	データなし (スキーマ定義のみ)
-o *filename*	ファイルに出力メッセージのログを取る
-p *char*	外部アンロードのためのエスケープ文字を指定する (デフォルトは "¥")
-q	クワイエット・モード (ウィンドウまたはメッセージは表示されない)
-r *reload-file*	生成された再ロード Interactive SQL コマンド・ファイルの名前とディレクトリを指定する (デフォルトは *reload.sql*)
-t *table,…*	リストされたテーブルだけをアンロードする
-u	データを順序付けしない。データのアンロードにインデックスを使用しない。
-v	冗長メッセージ
-xi	外部アンロード、内部再ロード
-xx	外部アンロード、外部再ロード
-y	確認メッセージを表示せずにコマンド・ファイルを置き換える

説明

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。

このモードでファイル名とパスを指定する方法については、『ASA SQL リファレンス・マニュアル』> 「UNLOAD TABLE 文」を参照してください。

-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。

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。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

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。詳細については、『SQL Remote ユーザーズ・ガイド』> 「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」を参照してください。

パスワードの大文字と小文字の区別

-an または -ar を使用してデータベースを再ロードする場合、パスワードの大文字小文字を区別する設定が保持されます。たとえば、大文字と小文字を区別するパスワードを持つデータベースをアンロードして -an または -ar を指定した場合、新規作成されたデータベースも大文字小文字を区別するパスワードを持つことになります。-an または -ar を指定しない場合、パスワードの大文字と小文字の区別は次のように決まります。

- 最初に大文字と小文字を区別するパスワードを使用したデータベースへパスワードを入力した場合、大文字と小文字を区別しないデータベースに再ロードされた場合でも、パスワードには大文字と小文字の区別が残ります。
- 最初に大文字と小文字を区別しないパスワードを使用したデータベースへパスワードを入力した場合、大文字と小文字を区別するデータベースへパスワードを再ロードするときは、パスワードを小文字で入力してください。

暗号化されたデータベース

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=dbeng9 -ep"
```

dbunload –an を使用してデータベースをアンロードして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。

暗号化の詳細については、「-ep サーバ・オプション」184 ページと「DatabaseKey 接続パラメータ [DBKEY]」250 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には、データベースを使ってデータベース・サーバを起動し、DBA ユーザ ID とパスワードを使ってアンロード・ユーティリティを実行します。

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。

Windows 95/98/Me、Windows NT/2000/XP、UNIX 環境では、アンロードと再ロードの処理を自動化するファイル (*rebuild.bat*、*rebuild.cmd*、または *rebuild*) があります。

詳細については、「再構築ユーティリティ」734 ページを参照してください。

アンロード・ユーティリティのオプション

@data　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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

データベースを再ロードするための接続パラメータ (-ac)　このオプションを指定すると、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します。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処理と、既存データベースへの結果の再ロード処理を組み合わせることができます。

通常は、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します。この方法は、ページ・サイズや照合などの初期化オプションを変更したい場合に便利です。照合を変更する場合、-xx オプションも使用して、古いデータベースと新しいデータベースの両方で適切に文字セット変換が行われているかを確認してください。

たとえば、次のコマンド (すべて 1 行に入力) は、*asademo.db* データベースのコピーを *newdemo.db* という名前の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=asademo.db" -ac
"uid=DBA;pwd=SQL;dbf=newdemo.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。

再ロード用のデータベースを作成する (-an) 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、パーソナル・サーバ接続と共有メモリを介したネットワーク・サーバ接続に適用されます。-an を指定する場合、パスワードの大文字と小文字を区別する設定は保持されます。

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、*asacopy.db* という名前の新規のデータベース・ファイルを作成し、*asademo.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=asademo.db" -an
asacopy.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は高まりますが、パフォーマンスは多少悪くなります。

このオプションではディレクトリを指定する必要はありません。

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。

新規データベース用にページ・サイズを設定する (-ap) このオプションを使用して、新規データベースのページ・サイズを設定できます。データベースのページ・サイズには、1024、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 2048

バイトです。このオプションと共に -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。

詳細については、「-gp サーバ・オプション」195 ページを参照してください。

データベースを再構築して置き換える (-ar)　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。-an を指定する場合、パスワードの大文字と小文字を区別する設定は保持されます。

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。

トランザクション・ログの管理については、「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」502 ページを参照してください。

ソース・データベース用の接続パラメータ (-c)　接続パラメータの詳細については、「接続パラメータ」236 ページを参照してください。ユーザ ID に DBA 権限を持たせて、データベースのすべてのテーブル上でユーザがパーミッションを保有するようにしてください。

たとえば、次の文は **sample_server** サーバ上で実行されるデータベース **asademo** をアンロードして、パスワード **SQL** を持つユーザ ID **DBA** として接続を行います。データは *c:¥unload* ディレクトリへアンロードされます。

```
dbunload -c
"eng=sample_server;dbn=asademo;uid=DBA;pwd=SQL"
c:¥unload
```

データのみアンロードする (-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。

指定されているテーブルのデータを出力しない (-e) このオプションは、このユーティリティをコマンド・プロンプトで実行している場合にのみ利用できます。データベース内のほとんどすべてのテーブルをアンロードする場合、-e オプションを指定すると、指定したテーブル以外のすべてのテーブルがアンロードされます。−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。

暗号化アルゴリズムを指定する (-ea) このオプションにより、新しいデータベースの暗号化に使用する強力な暗号化アルゴリズムを選択できます。AES (デフォルト) または AES_FIPS (FIPS 承認のアルゴリズムの場合) を選択できます。AESS_FIPS は個別のライブラリを使用するため、AES との互換性はありません。アルゴリズム名には大文字と小文字の区別はありません。-ea 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。-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。

強力なデータベース暗号化の詳細については、『SQL Anywhere Studio セキュリティ・ガイド』> 「高度な暗号化」を参照してください。

別途ライセンスを取得できるオプションが必要
AES_FIPS を使用した強力なデータベース暗号化には、個別にライセンスを取得可能な SQL Anywhere Studio セキュリティ・オプションを入手する必要があります。このセキュリティ・オプションは、輸出規制対象品目です。

このコンポーネントを注文するには、『SQL Anywhere Studio の紹介』> 「別途ライセンスが入手可能なコンポーネント」を参照してください。

暗号化キーを指定する (-ek)　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。

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。

強力なデータベース暗号化の詳細については、『SQL Anywhere Studio セキュリティ・ガイド』> 「高度な暗号化」を参照してください。

暗号化キーを入力するよう要求する (-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。

強力なデータベース暗号化の詳細については、『SQL Anywhere Studio セキュリティ・ガイド』> 「高度な暗号化」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。

アンロード・ユーティリティのファイル名とパスの詳細については、『ASA SQL リファレンス・マニュアル』>「UNLOAD TABLE 文」を参照してください。

内部アンロード、内部再ロードを使用する (-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。

内部アンロード、外部再ロードを使用する (-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。

Java を無視する (-jr) このオプションを指定すると、データベースのアンロード時または再ロード時に Java が無視されます。

-m このオプションを指定すると、データベースのレプリケーションでユーザ ID が保存されません。

スキーマ定義のみアンロードする (-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。

ファイルに出力メッセージのログを取る (-o) 指定したファイルに、出力メッセージを書き込みます。

エスケープ文字 (-p) 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) に使用するデフォルトのエスケープ文字 (¥) を別の文字に置き換えることができます。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。

クワイエット・モードで処理を実行する (-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。-q を指定する場合、-y も指定しないとアンロードは失敗します。

再ロード・ファイル名を指定する (-r)　生成される再ロード Interactive SQL コマンド・ファイルの名前とディレクトリを変更します。デフォルトは現在のディレクトリの中の *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。

リストされたテーブルだけをアンロードする (-t)　アンロードするテーブルのリストを指定します。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。

順序付けすることなくデータを出力する (-u)　通常、それぞれのテーブルのデータはプライマリ・キーによって順序付けられます。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。

冗長モードを有効にする (-v)　現在アンロード中のテーブル名とアンロードされたローの数を表示します。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。

外部アンロード、内部再ロードを使用する (-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します。

外部アンロード、外部再ロードを使用する (-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します。

確認メッセージを表示することなく処理を実行する (-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 も指定しないとアンロードは失敗します。

アップグレード・ユーティリティ

以前のバージョンの Adaptive Server Anywhere で作成したデータベースは、それよりも新しいリリースの Adaptive Server Anywhere で実行できますが、新しい機能を使用するにはデータベースをアップグレードする必要があります。

アップグレード・ユーティリティを使用すると、データベースが古いバージョンの Adaptive Server Anywhere から新しいバージョンにアップグレードされ、新しいリリースの一連の機能を利用できるようになります。

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。

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。
- InteractiveSQL の ALTER DATABASE UPGRADE 文を使用する。

  詳細については、『ASA SQL リファレンス・マニュアル』> 「ALTER DATABASE 文」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。

[データベース・アップグレード] ウィザードを使用したデータベースのアップグレード

❖ データベースをアップグレードするには、次の手順に従います。

1　データベースに接続します。

2　左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。

3　右ウィンドウ枠にある [ユーティリティ] タブをクリックします。

4　右ウィンドウ枠の [データベースのアップグレード] をダブルクリックします。

[データベース・アップグレード] ウィザードが表示されます。

5　ウィザードの指示に従います。

ヒント

[データベース・アップグレード] ウィザードは、次の方法でもアクセスできます。

- [ツール] － [Adaptive Server Anywhere 9] － [データベースのアップグレード] 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 [データベースのアップグレード] を選択する。
- 左ウィンドウ枠でデータベースを選択し、[ファイル] － [データベースのアップグレード] を選択する。

古くてアップグレード・ユーティリティではアップグレードできないデータベースをアップグレードする

❖ **古くてアップグレードできない Adaptive Server Anywhere のバージョンでデータベースが作成されている場合にアップグレードするには、次の手順に従います。**

1. 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ます。
2. 初期化ユーティリティを使用して、アップグレードされたバージョンに使用する名前でデータベースを作成します。
3. DBA ユーザ ID として、Interactive SQL から新しいデータベースに接続し、*reload.sql* コマンド・ファイルを読み込んで新しいデータベースを構築します。

dbunload ユーティリティを使用して、直接再構築することもできます。

dbunload ユーティリティの使用については、「データベースの再構築」 768 ページを参照してください。

dbupgrad コマンド・ライン・ユーティリティを使用したデータベースのアップグレード

構文 **dbupgrad** [*options*]

オプション	説明
@data	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出す
-c *"keyword=value; …"*	データベース接続パラメータを指定する
-l	Sybase jConnect サポートをインストールしない
-j	ランタイム Java クラスをアップグレードしない

オプション	説明
-ja	ランタイム Java クラスを追加する
-jdk *version*	Java Development Kit の指定したバージョンのエントリをインストールする
-jr	ランタイム Java クラスを削除する
-o *filename*	ファイルに出力メッセージのログを取る
-q	クワイエット・モード (ウィンドウまたはメッセージは表示されない)

説明

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 Watcom SQL 3.2 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。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

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。

詳細については、『SQL Anywhere Studio 新機能ガイド』> 「データベース・ファイル・フォーマットのアップグレード」を参照してください。

データベース内の Java

デフォルトでは、データベース内の Java は、新しいデータベースには含まれません。データベース内の Java は、次の条件がすべて満たされている場合は、アップグレード中に削除されます。

- データベースは Java 対応になっているが、データベース内の Java は使用されていない。
- Java VM がデータベース・サーバによってロードできない。
- アップグレードに対して Java 関連オプションを指定しなかった。

ランタイム・クラスをインストールすると、データベースのサイズが数メガバイト増加します。Java クラスを使用する予定がない場合は、-j オプションを指定して、アップグレードするデータベースの中にこれらのクラスを入れないようにすることができます。必要であれば、Sybase Central または ALTER DATABASE 文を使用して、Sybase ランタイム Java クラスを後から追加できます。

詳細については、『ASA プログラミング・ガイド』> 「データベースを Java 実行可能にする」を参照してください。

データベース内の Java は、別途ライセンスが入手可能なコンポーネントです。これらのクラスは、データベース内の Java オプションをインストールしている場合、またはアップグレードするデータベース内で Java クラスが使用されていた場合のみ、アップグレード中にインストールされます。

アップグレード・ユーティリティのオプション

@data 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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

接続パラメータ (-c) 接続パラメータの詳細については、「接続パラメータ」236 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 sample80 という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、それをバージョン 9 フォーマットにアップグレードします。

```
dbupgrad -c "uid=DBA;pwd=SQL;dbf=c:¥asa80¥sample80.db"
```

dbupgrad ユーティリティは、DBA 権限のあるユーザが実行してください。

Sybase jConnect サポートをインストールしない (-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。

詳細については、『ASA プログラミング・ガイド』> 「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」を参照してください。

ランタイム Java クラスをインストールしない (-j)　Java クラスを使用しない場合は、Java オプションを指定しないか、-j オプションを指定して、これらのクラスをデータベースに組み込まないようにできます。

ランタイム Java クラスをインストールする (-ja)　アップグレードしたデータベースに、Java を追加します。-ja を指定すると、バージョン 1.3 の JDK がデータベースにインストールされます。

JDK の指定バージョンのサポートをインストールする (-jdk)　特定のバージョンの Java をインストールする場合は、-jdk オプションの後にバージョン番号を指定します。有効な値は、1.1.8 と 1.3 です。

たとえば、次のコマンドは、データベース内で JDK 1.1.8 アプリケーションをサポートするデータベースをアップグレードします。

```
dbupgrad -jdk 1.1.8 java2.db
```

-jdk オプションは、-ja オプションも暗黙で指定したことになります。

ランタイム Java クラスを削除する (-jr) データベース内の Java を、アップグレードしたデータベースから削除します。

ファイルに出力メッセージのログを取る (-o) 指定したファイルに、出力メッセージを書き込みます。

クワイエット・モードで処理を実行する (-q) 出力メッセージを表示しません。

検証ユーティリティ

検証ユーティリティを使用すると、データベース内の一部またはすべてのテーブルについて、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は全テーブルをスキャンし、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。

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 (「バックアップとデータ・リカバリ」483 ページを参照)、データベースのデータの整合性を保持できます。

次の方法で、検証ユーティリティにアクセスできます。

- Sybase Central の [データベース検証] ウィザードを使用する。
- InteractiveSQL の sa_validate ストアド・プロシージャまたは CREATE DATABASE 文を使用する。
- コマンド・プロンプトで、dbvalid 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。

テーブルの検証の詳細については、『ASA SQL リファレンス・マニュアル』> 「VALIDATE TABLE 文」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。そうでない場合、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。

[データベース検証] ウィザードを使用したデータベースの検証

Sybase Central からデータベース、または個々のテーブルを検証できます。

❖ データベースを検証するには、次の手順に従います。

1 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。

2 右ウィンドウ枠にある [ユーティリティ] タブをクリックします。

3 右ウィンドウ枠で、[データベースの検証] をダブルクリックします。

[データベース検証] 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。

❖ 実行中のデータベースを検証するには、次の手順に従います。

1 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。

❖ 個々のテーブルを検証するには、次の手順に従います。

1 データベースに接続します。

2 検証するテーブルを見つけます。

3 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。

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。

- [ツール] ー [Adaptive Server Anywhere 9] ー [データベースの検証] 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 [データベースの検証] を選択する。
- 左ウィンドウ枠でデータベースを選択し、[ファイル] ー [データベースの検証] を選択する。

dbvalid コマンド・ライン・ユーティリティを使用したデータベースの検証

構文　**dbvalid** [*options*] [*object-name, …*]

オプション	説明
@data	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出す
-c *"keyword=value; …"*	データベース接続パラメータを指定する
-f	テーブルをフル・チェックで検証する
-fd	テーブルをデータ・チェックで検証する
-fi	テーブルをインデックス・チェックで検証する
-fn	テーブルを従来の検証アルゴリズムで検証する (バージョン 9.0.0 以前のソフトウェアで使用)
-fx	テーブルをエクスプレス・チェックで検証する
-l	各 *object-name* はインデックス
-o *filename*	ファイルに出力メッセージのログを取る
-q	クワイエット・モード (メッセージを表示しない)
-s	チェックサムを使用してデータベース・ページを検証する
-t	各 *object-name* はテーブル

オプション	説明
object-name	検証するテーブル名、またはインデックス名

説明

検証ユーティリティを使用すると、データベース内の一部またはすべてのテーブルについて、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はテーブル全体をスキャンし、テーブルのローが適切なインデックス内に存在することを確認します。各テーブル上で、VALIDATE TABLE 文を実行するのと同じ効果があります。

デフォルトでは、検証ユーティリティはエクスプレス・チェック・オプションを使用します。ただし、このエクスプレス・チェック・オプションは、−f、−fd、−fi、−fn、−I を指定した場合、またはデータベースがバージョン 7.0 以前のソフトウェアで作成された場合には*使用されません*。

終了コードは次のとおりです。

プログラム終了コード	説明
0	データベースの検証に成功
1	ユーティリティで一般的な失敗
2	データベースの検証エラー
7	接続するデータベースが見つけられない (データベース名が間違っている)
8	接続するデータベースが見つけられない (ユーザ ID ／パスワードが間違っている)
11	接続するサーバが見つけられない (サーバ名が間違っている)
12	データベース起動用の暗号化キーが間違っている

検証中に行われる特定のチェックについては、『ASA SQL リファレンス・マニュアル』> 「VALIDATE TABLE 文」を参照してください。

検証ユーティリティのオプション

@data 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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

接続パラメータ (-c)　接続パラメータの詳細については、「接続パラメータ」236 ページを参照してください。DBA 権限または REMOTE DBA 権限 (SQL Remote) のあるユーザ ID を使用してください。

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・データベースを検証します。

```
dbvalid -c "uid=DBA;pwd=SQL;dbf=c:¥asa¥asademo.db"
```

各テーブルのフル・チェック (-f)　デフォルトの検証チェックに加えて、データ・チェック (-fd) とインデックス・チェック (-fi) の両方を実行します。これは、VALIDATE TABLE 文の WITH DATA CHECK オプションに対応しています。データベースの内容によっては、このオプションによる検証時間が大幅に長くなることがあります。

各テーブルのデータ・チェック (-fd)　デフォルトの検証チェックに加えて、すべての LONG BINARY、LONG VARCHAR、TEXT、IMAGE データ・タイプがそれぞれ読み込み可能かどうかをチェックします。これは、VALIDATE TABLE 文の WITH DATA CHECK オプションに対応しています。データベースの内容によっては、このオプションによる検証時間が大幅に長くなることがあります。

各テーブルのインデックス・チェック (-fi)　デフォルトの検証チェックに加えて、テーブルの各インデックスを検証します。これは、VALIDATE TABLE 文の WITH DATA CHECK オプションに対応しています。データベースの内容によっては、このオプションによる検証時間が大幅に長くなることがあります。

各テーブルのチェックに従来の検証アルゴリズムを使用する (-fn)　エクスプレス・チェック・アルゴリズムではなく、バージョン 9.0.0 以前のソフトウェアで使用されていた検証アルゴリズムを使用します。

各テーブルのエクスプレス・チェック (-fx) デフォルト・チェックとデータ・チェック (-fd) に加え、テーブル内のローの数とインデックス内のエントリの数が一致しているかどうかをチェックします。これは、VALIDATE TABLE STATEMENT の WITH EXPRESS CHECK オプションに対応しています。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。エクスプレス・チェック・オプションは、デフォルトでオンになっています。

指定のインデックスを検証する (-I) テーブルを検証せず、インデックスを検証します。この場合、dbvalid では、指定されるそれぞれの *object-name* 値は、テーブルではなくインデックスを表し、次のような形式の名前を持ちます。

[[*owner*.]*table-name*.]*index-name*

ファイルに出力メッセージのログを取る (-o) 指定したファイルに、出力メッセージを書き込みます。

クワイエット・モードで処理を実行する (-q) 出力メッセージを表示しません。

ページのチェックサムを使用してデータベースを検証する (-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、-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。

テーブルを検証する (-t) object-name 値のリストは、テーブルのリストを表します。これはデフォルトの動作でもあります。

object-name 検証するテーブルの名前。

-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。

ライト・ファイル・ユーティリティ (旧式)

ライト・ファイル・ユーティリティを使って、データベースのライト・ファイルを管理します。「**ライト・ファイル**」は特定のデータベースに添付されているファイルです。すべての変更内容はライト・ファイルに書き込まれ、データベース・ファイルは変更されません。

推奨されなくなった機能
ライト・ファイルは推奨されなくなりました。

開発を目的としたライト・ファイルの使い方

ライト・ファイルは、運用データベースを修正したくないときに、テスト用として効果的に使用できます。ライト・ファイルは、データベースへの読み込み専用アクセスが必要なネットワーク環境で使用したり、ユーザによる修正が可能なデータベースを CD-ROM で配布する場合にも使用できます。

圧縮データベース

圧縮データベースを使っている場合は、ライト・ファイルを使用してください。圧縮データベース・ファイルは、直接アクセスできません。データベースに接続したり、データベース・サーバのコマンド・ラインにデータベースをロードする場合は、データベース名の代わりにライト・ファイル名を使用できます。

次の方法で、ライト・ファイル・ユーティリティにアクセスできます。

- Sybase Central の [ライト･ファイル作成] ウィザードを使用する。
- コマンド・プロンプトで、dbwrite 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。

ライト・ファイル・ユーティリティは、データベース・ファイルに対して実行されます。ライト・ファイル・ユーティリティを起動する場合には、データベースが起動していないことを確認してください。

ライト・ファイルの作成後、データベース・ファイルの修正はできません。データ・ファイルを修正すると、ライト・ファイルは無効になります。

[ライト・ファイル作成] ウィザードを使用したライト・ファイルの作成

❖ **データベースにライト・ファイルを作成するには、次の手順に従います。**

1. 左ウィンドウ枠で、Adaptive Server Anywhere プラグインを選択します。
2. 右ウィンドウ枠にある [ユーティリティ] タブをクリックします。
3. 右ウィンドウ枠で、[ライト・ファイルの作成] をダブルクリックします。

   [ライト・ファイル作成] 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

Sybase Central では、次の方法で [ライト・ファイル作成] ウィザードを利用することもできます。

- [ツール] ー [Adaptive Server Anywhere 9] ー [ライト・ファイルの作成] を選択する。
- サーバを右クリックし、ポップアップ・メニューから [ライト・ファイルの作成] を選択する。
- 左ウィンドウ枠でサーバを選択し、[ファイル] ー [ライト・ファイルの作成] を選択する。

dbwrite コマンド・ライン・ユーティリティを使用したライト・ファイルの作成

構文 **dbwrite** [*options*] *database-file* [*write-name*]

オプション	説明
@data	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出す
-c	新しいライト・ファイルを作成する
-d *database-file*	ライト・ファイルを別のデータベースにポイントする
-ek *key*	暗号化キーを指定する
-ep	暗号化キーを入力するよう要求する
-f *database-file*	ライト・ファイルにファイルを指定させる
-m *mirror-name*	トランザクション・ログ・ミラー名を設定する
-o *filename*	ファイルに出力メッセージのログを取る
-q	クワイエット・モード (メッセージを表示しない)
-s	ライト・ファイル状態を通知する
-t *log-name*	トランザクション・ログ名を設定する
-y	確認メッセージを表示せずにファイルを消去する

説明

元のデータベースに (ライト・ファイルを使わずに) 変更を加えると、ライト・ファイルは有効ではなくなります。元のデータベース・ファイルを使用してサーバを起動すると、ライト・ファイルが無効になるため、データベースのアーカイブされたコピーからライト・ファイルを作成してください。

ライト・ファイルが無効になった場合、次のコマンドによって、ライト・ファイルを再度有効にできます。ただし、ライト・ファイルに記録されているすべての変更は削除されます。

```
dbwrite -c db-name  write-name
```

新しいライト・ファイルを作成するときにのみ、*log-name* と *mirror-name* パラメータを使います。*write-name* パラメータは、−c と −d パラメータとだけ使用します。*database-file* パラメータは、*write-name* パラメータの前に指定してください。

終了コードは、0 (成功) または 0 以外の値 (失敗) です。

ライト・ファイル・ユーティリティのオプション

@data 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オプションを読み出します。両方が同じ名前で存在する場合は、環境変数値が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」642 ページを参照してください。

設定ファイルに含まれる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して、設定ファイルの内容を難読化できます。

詳細については、「dbfhide コマンド・ライン・ユーティリティを使用してファイル内容を隠す」679 ページを参照してください。

新しいライト・ファイルを作成する (-c) 既存のライト・ファイルがある場合、古いライト・ファイルの情報は失われます。コマンドにライト・ファイル名を指定しない場合、ライト・ファイル名は、デフォルトでデータベース名に拡張子 *wrt* が付いたものになります。トランザクション・ログ名を指定しない場合、ログ・ファイル名は、デフォルトでデータベース名に拡張子 *wlg* が付いたものになります。

既存のライト・ファイルが指定するデータベース・ファイルを変更する (-d) このオプションを使用すると、データベース・ファイルを別のディレクトリに移動したり名前を変更したりした場合でも、ライト・ファイルとデータベース・ファイル間のリンクを維持できます。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。

暗号化キーを指定する (-ek) 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。強力に暗号化されたデータベースの場合、-ek または -ep をどちらか指定します。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。

暗号化キーを入力するよう要求する (-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。強力に暗号化されたデータベースの場合、-ek または -ep をどちらか指定します。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。

ライト・ファイルにファイルを指定させる (-f)　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。このオプションを使うのは、直接入力できないオペレーティング・システムの場合で、ライト・ファイルが作成されていて、データベース・ファイルが Novell NetWare または他のネットワーク・パス上に格納されるときです。データベース・ファイルに Novell のフル・パス名 (たとえば、*SYS:¥asademo.db*) を指定することで、NetWare パスのローカル・マッピングへの依存を避けることができます。-d オプションとは異なり、指定されたパス上でのチェックは行われません。

ミラー名を設定する (-m)　トランザクション・ログ・ミラー・ファイルの名前を設定します。これは、-c と組み合わせた場合のみ使用できます。

ファイルに出力メッセージのログを取る (-o)　指定したファイルに、出力メッセージを書き込みます。

クワイエット・モードで処理を実行する (-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。

ライト・ファイルのステータスのみをレポートする (-s)　ライト・ファイルが指定するデータベースの名前を表示します。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。

トランザクション・ログ・ファイル名を設定する (-t)　新しいライト・ファイルを作成する場合 (-c)、このオプションを使用して、トランザクション・ログ・ファイルの名前を設定できます。

確認メッセージを表示することなく処理を実行する (-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、既存のライト・ファイルが置き換えられます。

第 16 章

データベース・オプション

この章の内容

この章では、データベースの動作をカスタマイズしたり修正するために設定できるデータベース、レプリケーション、互換性、Interactive SQL の各オプションについて説明します。

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** 使用している Adaptive Server Anywhere データベースがどの程度 Adaptive Server Enterprise と同様の操作をするか、また SQL が SQL/92 に準拠しない場合にエラーを発生させるかどうかを制御できます。
- **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。

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 = mixed
```

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。

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、明確に定義されていない動作を引き起こすことがあるので、サポートされていません。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。

詳細については、『ASA SQL リファレンス・マニュアル』>「SET OPTION 文」を参照してください。

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
  ```

  sa_conn_properties_by_name を呼び出して、このプロパティを並べ替えることもできます。

  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
  ```

詳細については、『ASA SQL リファレンス・マニュアル』> 「sa_conn_properties_by_name システム・プロシージャ」と 『ASA SQL リファレンス・マニュアル』> 「sa_conn_properties システム・プロシージャ」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 では、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 [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')
```

詳細については、『ASA SQL リファレンス・マニュアル』> 「CONNECTION_PROPERTY 関数 [システム]」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に優先します。ユーザ・レベルのオプションは、パブリック設定に優先します。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、次にサーバ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。

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 のオプションを設定すると、一時的にセキュリティの利点を提供します。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。このオプションを一時的に有効にすると、Windows NT/2000/XP 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・マシン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。

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 のオプション値を変更すると、独自の数値を設定していない全ユーザのオプション値を変更することになります。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 ユーザに限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。

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 =
```

詳細については、『ASA SQL リファレンス・マニュアル』>「SET OPTION 文」を参照してください。

オプション分類

Adaptive Server Anywhere には多くのオプションがあります。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」802 ページ

- 「互換性オプション」809 ページ
- 「レプリケーション・オプション」814 ページ
- 「Interactive SQL オプション」816 ページ

オプションの初期設定

Adaptive Server Anywhere への接続は、TDS プロトコル (Open Client と jConnect JDBC 接続) または Adaptive Server Anywhere プロトコル (ODBC と Embedded SQL) を使用して実行できます。

TDS プロトコルと Adaptive Server Anywhere 固有のプロトコルの両方を使用するユーザの場合、初期設定はストアド・プロシージャを使用して行えます。Adaptive Server Anywhere の出荷時には、このメソッドを使用して、デフォルトの Adaptive Server Enterprise の動作を反映するように、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。必要であれば、この動作は変更できます。

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。

詳細については、「LOGIN_PROCEDURE オプション [データベース]」863 ページ、『ASA SQL リファレンス・マニュアル』> 「sp_login_environment システム・プロシージャ」、および 『ASA SQL リファレンス・マニュアル』> 「sp_tsql_environment システム・プロシージャ」を参照してください。

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
「AUDITING オプション [データベース]」 824 ページ	ON、OFF	OFF
「BACKGROUND_PRIORITY オプション [データベース]」 826 ページ	ON、OFF	OFF
「BLOCKING オプション [データベース]」 828 ページ	ON、OFF	ON
「BLOCKING_TIMEOUT オプション [データベース]」 828 ページ	整数	0
「CHECKPOINT_TIME オプション [データベース]」 829 ページ	分	60
「CIS_OPTION オプション [データベース]」 830 ページ	0, 7	0
「CIS_ROWSET_SIZE オプション [データベース]」 831 ページ	整数	50
「COOPERATIVE_COMMIT_TIMEOUT オプション [データベース]」 835 ページ	整数	250
「COOPERATIVE_COMMITS オプション [データベース]」 835 ページ	ON、OFF	ON
「DEBUG_MESSAGES オプション [データベース]」 839 ページ	ON、OFF	OFF
「DEDICATED_TASK オプション [データベース]」 840 ページ	ON、OFF	OFF

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]」 841 ページ	整数	1
「DELAYED_COMMIT_TIMEOUT オプション [データベース]」 842 ページ	整数	500
「DELAYED_COMMITS オプション [データベース]」 843 ページ	ON、OFF	OFF
「EXCLUDE_OPERATORS オプション [データベース]」 846 ページ	予約	予約
「EXTENDED_JOIN_SYNTAX オプション [データベース]」 846 ページ	ON、OFF	ON
「FIRST_DAY_OF_WEEK オプション [データベース]」 847 ページ	1, 2, 3, 4, 5, 6, 7	7
「FOR_XML_NULL_TREATMENT オプション [データベース]」 849 ページ	EMPTY、OMIT	OMIT
「FORCE_VIEW_CREATION オプション [データベース]」 850 ページ	予約	予約
「GLOBAL_DATABASE_ID オプション [データベース]」 850 ページ	整数	2147483647
「INTEGRATED_SERVER_NAME オプション [データベース]」 853 ページ	文字列	NULL

オプション	値	デフォルト
「JAVA_HEAP_SIZE オプション [データベース]」860 ページ	バイト数	1 000 000
「JAVA_INPUT_OUTPUT オプション [データベース]」860 ページ	ON、OFF	OFF
「JAVA_NAMESPACE_SIZE オプション [データベース]」861 ページ	バイト数	4 000 000
「LOG_DEADLOCKS オプション [データベース]」861 ページ	ON、OFF	OFF
「LOGIN_MODE オプション [データベース]」862 ページ	STANDARD、MIXED、INTEGRATED	STANDARD
「LOGIN_PROCEDURE オプション [データベース]」863 ページ	文字列	sp_login_environment
「MAX_CURSOR_COUNT オプション [データベース]」866 ページ	整数	50
「MAX_HASH_SIZE オプション [データベース] (旧式)」866 ページ	整数	10
「MAX_PLANS_CACHED オプション [データベース]」867 ページ	整数	20
「MAX_RECURSIVE_ITERATIONS オプション [データベース]」868 ページ	整数	100
「MAX_STATEMENT_COUNT オプション [データベース]」868 ページ	0 以上の整数	50

オプション	値	デフォルト
「MAX_WORK_TABLE_HASH_SIZE オプション [データベース] (旧式)」 869 ページ	整数	20
「MIN_PASSWORD_LENGTH オプション [データベース]」 870 ページ	整数	0
「ODBC_DESCRIBE_BINARY_AS_VARBINARY [データベース]」 872 ページ	ON、OFF	OFF
「ODBC_DISTINGUISH_CHAR_AND_VARCHAR オプション [データベース]」 873 ページ	ON、OFF	OFF
「ON_CHARSET_CONVERSION_FAILURE オプション [データベース]」 874 ページ	IGNORE、WARNING、ERROR	IGNORE
「OPTIMIZATION_GOAL オプション [データベース]」 877 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]」 878 ページ	0-15	9
「OPTIMIZATION_WORKLOAD オプション [データベース]」 879 ページ	Mixed、OLAP	Mixed
「PINNED_CURSOR_PERCENT_OF_CACHE オプション [データベース]」 884 ページ	0-100	10
「PRECISION オプション [データベース]」 884 ページ	桁数	30

オプション	値	デフォルト
「PREFETCH オプション [データベース]」 885 ページ	OFF、CONDITIONAL、ALWAYS	CONDITIONAL
「PRESERVE_SOURCE_FORMAT オプション [データベース]」 886 ページ	ON、OFF	ON
「PREVENT_ARTICLE_PKEY_UPDATE オプション [データベース]」 887 ページ	ON、OFF	ON
「READ_PAST_DELETED オプション [データベース]」 890 ページ	ON、OFF	ON
「RECOVERY_TIME オプション [データベース]」 890 ページ	分	2
「REMOTE_IDLE_TIMEOUT オプション [データベース]」 891 ページ	秒数	15
「RETURN_DATE_TIME_AS_STRING オプション [データベース]」 892 ページ	ON、OFF	OFF
「RETURN_JAVA_AS_STRING オプション [データベース]」 893 ページ	ON、OFF	OFF
「ROLLBACK_ON_DEADLOCK [データベース]」 894 ページ	ON、OFF	ON
「ROW_COUNTS オプション [データベース]」 895 ページ	ON、OFF	OFF
「SCALE オプション [データベース]」 895 ページ	桁数	6

オプション	値	デフォルト
「SORT_COLLATION オプション [データベース]」 896 ページ	有効な collation_name または collation_ID	Internal
「SUBSUME_ROW_LOCKS オプション [データベース]」 899 ページ	ON、OFF	ON
「SUPPRESS_TDS_DEBUGGING オプション [データベース]」 900 ページ	ON、OFF	OFF
「TDS_EMPTY_STRING_IS_NULL オプション [データベース]」 900 ページ	ON、OFF	OFF
「TEMP_SPACE_LIMIT_CHECK オプション [データベース]」 901 ページ	ON、OFF	OFF
「TIME_ZONE_ADJUSTMENT オプション [データベース]」 903 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) を示す文字列のいずれか	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_DATE_VALUES オプション [データベース] (旧式)」 905 ページ	ON、OFF	ON (バージョン 7 以降のデータベース) OFF (バージョン 7 より前のデータベース)
「TRUNCATE_TIMESTAMP_VALUES オプション [データベース]」 906 ページ	ON、OFF	OFF

オプション	値	デフォルト
「TRUNCATE_WITH_AUTO_COMMIT オプション [データベース]」 907 ページ	ON、OFF	OFF
「UPDATE_STATISTICS オプション [データベース]」 909 ページ	ON	ON
「USER_ESTIMATES オプション [データベース]」 910 ページ	ENABLED、OVERRIDE-MAGIC、DISABLED	OVERRIDE-MAGIC
「WAIT_FOR_COMMIT オプション [データベース]」 911 ページ	ON、OFF	OFF
「WEBSERVICE_NAMESPACE_HOST オプション [データベース]」 912 ページ	文字列、NULL	NULL

互換性オプション

次のオプションを使用すると、Adaptive Server 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/92 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、Adaptive Server 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することができます。

リストについては、『ASA SQL リファレンス・マニュアル』>「SET 文 [T-SQL]」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。互換性を保つためには、互換性のある動作を確保する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。

設定方法の詳細については、『ASA SQL リファレンス・マニュアル』>「システム・プロシージャとカタログ・ストアド・プロシージャ」を参照してください。

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	ON
ANSINULL	OFF
CHAINED	OFF
CONTINUE_AFTER_RAISERROR	ON
DATE_FORMAT	YYYY-MM-DD

オプション	設定
DATE_ORDER	MDY
ESCAPE_CHARACTER	OFF
FLOAT_AS_DOUBLE	ON
ISOLATION_LEVEL	1
ON_TSQL_ERROR	CONDITIONAL
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

Transact-SQL と SQL/92 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]」819 ページ	ON、OFF	ON
「ANSI_BLANKS オプション [互換性]」819 ページ	ON、OFF	OFF
「ANSI_CLOSE_CURSORS_ON_ROLLBACK オプション [互換性]」820 ページ	ON、OFF	OFF
「ANSI_INTEGER_OVERFLOW オプション [互換性]」820 ページ	ON、OFF	OFF
「ANSI_PERMISSIONS オプション [互換性]」821 ページ	ON、OFF	ON

オプション	値	デフォルト
「ANSI_UPDATE_CONSTRAINTS オプション [互換性]」822 ページ	OFF、CURSORS、STRICT	CURSORS
「ANSINULL オプション [互換性]」823 ページ	ON、OFF	ON
「AUTOMATIC_TIMESTAMP オプション [互換性]」826 ページ	ON、OFF	OFF
「CHAINED オプション [互換性]」829 ページ	ON、OFF	ON
「CLOSE_ON_ENDTRANS オプション [互換性]」831 ページ	ON、OFF	ON
「CONTINUE_AFTER_RAISERROR オプション [互換性]」833 ページ	ON、OFF	ON
「CONVERSION_ERROR オプション [互換性]」834 ページ	ON、OFF	ON
「DATE_FORMAT オプション [互換性]」836 ページ	*文字列*	YYYY-MM-DD
「DATE_ORDER オプション [互換性]」838 ページ	MDY、YMD、DMY	YMD
「DIVIDE_BY_ZERO_ERROR オプション [互換性]」845 ページ	ON、OFF	ON
「ESCAPE_CHARACTER オプション [互換性]」846 ページ	システム使用	OFF
「FIRE_TRIGGERS オプション [互換性]」847 ページ	ON、OFF	ON
「FLOAT_AS_DOUBLE オプション [互換性]」848 ページ	ON、OFF	OFF

オプション	値	デフォルト
「ISOLATION_LEVEL オプション [互換性]」853 ページ	0, 1, 2, 3	0 Open Client 接続と JDBC 接続の場合は 1
「NEAREST_CENTURY オプション [互換性]」870 ページ	整数	50
「NON_KEYWORDS オプション [互換性]」871 ページ	カンマで区切られたキーワード・リスト	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]」875 ページ	STOP、CONDITIONAL、CONTINUE	CONDITIONAL
「OPTIMISTIC_WAIT_FOR_COMMIT オプション [互換性]」876 ページ	ON、OFF	OFF
「PERCENT_AS_COMMENT オプション [互換性]」883 ページ	ON、OFF	ON
「QUERY_PLAN_ON_OPEN オプション [互換性]」888 ページ	ON、OFF	OFF
「QUOTED_IDENTIFIER オプション [互換性]」889 ページ	ON、OFF	ON
「RI_TRIGGER_TIME オプション [互換性]」894 ページ	BEFORE、AFTER	AFTER
「SQL_FLAGGER_ERROR_LEVEL オプション [互換性]」897 ページ	E、I、F、W	W
「SQL_FLAGGER_WARNING_LEVEL オプション [互換性]」897 ページ	E、I、F、W	W

オプション	値	デフォルト
「STRING_RTRUNCATION オプション [互換性]」898 ページ	ON、OFF	OFF
「TIME_FORMAT オプション [互換性]」902 ページ	文字列	HH:NN:SS.SSS
「TIMESTAMP_FORMAT オプション [互換性]」903 ページ	文字列	YYYY-MM-DD HH:NN:ss.SSS
「TSQL_HEX_CONSTANT オプション [互換性]」908 ページ	ON、OFF	ON
「TSQL_VARIABLES オプション [互換性]」909 ページ	ON、OFF	OFF

レプリケーション・オプション

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。レプリケーション・オプションには、SQL Remote レプリケーションに便利なものや、Sybase Replication Server との使用に関連のあるものがあります。

オプション	値	デフォルト
「BLOB_THRESHOLD オプション [レプリケーション]」827 ページ	整数	256
「COMPRESSION オプション [レプリケーション]」833 ページ	-1 ～ 9 の整数	6
「DELETE_OLD_LOGS オプション [レプリケーション]」844 ページ	ON、OFF	OFF
「QUALIFY_OWNERS オプション [レプリケーション]」888 ページ	ON、OFF	ON
「QUOTE_ALL_IDENTIFIERS オプション [レプリケーション]」888 ページ	ON、OFF	OFF
「REPLICATE_ALL オプション [レプリケーション]」891 ページ	ON、OFF	OFF
「REPLICATION_ERROR オプション [レプリケーション]」891 ページ	プロシージャ名	(プロシージャなし)
「REPLICATION_ERROR_PIECE オプション [レプリケーション]」892 ページ	プロシージャ名	(プロシージャなし)

オプション	値	デフォルト
「SUBSCRIBE_BY_REMOTE オプション [レプリケーション]」899 ページ	ON、OFF	ON
「VERIFY_ALL_COLUMNS オプション [レプリケーション]」911 ページ	ON、OFF	OFF
「VERIFY_THRESHOLD オプション [レプリケーション]」911 ページ	整数	1 000

Interactive SQL オプション

構文 1 **SET** [**TEMPORARY**] **OPTION**

[*userid*. | **PUBLIC**.]*option-name* = [*option-value*]

構文 2 **SET PERMANENT**

構文 3 **SET**

パラメータ *userid*: *identifier*、*string*、または *host-variable*

option-name: *identifier*、*string*、または *host-variable*

option-value: *host-variable* (**インジケータ使用可**)、*string*、*identifier*、または *number*

説明 SET PERMANENT (構文 2) は、すべての現在の Interactive SQL オプションを SYSOPTION システム・テーブルに格納します。Interactive SQL が現在のユーザ ID で起動するたびに、これらの設定が自動的に確立されます。

構文 3 を使用すると、現在のオプション設定がすべて表示されます。Interactive SQL またはデータベース・サーバに一時的に設定されているオプションがある場合は、それが表示されます。それ以外の場合、永久オプション設定が表示されます。

オプション	値	デフォルト
「AUTO_COMMIT オプション [Interactive SQL]」825 ページ	ON、OFF	OFF
「AUTO_REFETCH オプション [Interactive SQL]」826 ページ	ON、OFF	ON
「BELL オプション [Interactive SQL]」827 ページ	ON、OFF	ON

オプション	値	デフォルト
「COMMAND_DELIMITER オプション [Interactive SQL]」 832 ページ	文字列	';'
「COMMIT_ON_EXIT オプション [Interactive SQL]」 832 ページ	ON、OFF	ON
「DEFAULT_ISQL_ENCODING オプション [Interactive SQL]」 840 ページ	文字列	(空の文字列)
「DESCRIBE_JAVA_FORMAT オプション [Interactive SQL]」 844 ページ	Varchar、binary	Varchar
「ECHO オプション [Interactive SQL]」 845 ページ	ON、OFF	ON
「INPUT_FORMAT オプション [Interactive SQL]」 851 ページ	ASCII、DBASE、DBASEII、DBASEIII、EXCEL、FIXED、FOXPRO、LOTUS	ASCII
「ISQL_COMMAND_TIMING オプション [Interactive SQL]」 854 ページ	ON、OFF	ON
「ISQL_ESCAPE_CHARACTER オプション [Interactive SQL]」 855 ページ	文字	'¥'
「ISQL_FIELD_SEPARATOR オプション [Interactive SQL]」 856 ページ	文字列	','
「ISQL_LOG オプション [Interactive SQL]」 856 ページ	ファイル名	(空の文字列)

オプション	値	デフォルト
「ISQL_PLAN オプション [Interactive SQL]」857 ページ	NONE、SHORT、LONG、GRAPHICAL、GraphicalWithStatistics	SHORT
「ISQL_PRINT_RESULT_SET オプション [Interactive SQL]」858 ページ	LAST、ALL、NONE	LAST
「ISQL_QUOTE オプション [Interactive SQL]」859 ページ	文字列	'
「NULLS オプション [Interactive SQL]」872 ページ	文字列	'(NULL)'
「ON_ERROR オプション [Interactive SQL]」874 ページ	STOP、CONTINUE、PROMPT、EXIT、NOTIFY_CONTINUE、NOTIFY_STOP、NOTIFY_EXIT	PROMPT
「OUTPUT_FORMAT オプション [Interactive SQL]」880 ページ	ASCII、DBASEII、DBASEIII、EXCEL、FIXED、FOXPRO、HTML、LOTUS、SQL、XML、	ASCII
「OUTPUT_LENGTH オプション [Interactive SQL]」882 ページ	整数	0
「OUTPUT_NULLS オプション [Interactive SQL]」882 ページ	文字列	'NULL'
「STATISTICS オプション [Interactive SQL]」898 ページ	0,3,4,5,6	3
「TRUNCATION_LENGTH オプション [Interactive SQL]」908 ページ	整数	256

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]

機能　**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 JDBC 接続の場合は **OFF**

説明　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。

詳細については、『ASA SQL ユーザーズ・ガイド』>「Transact-SQL との互換性を維持するためのオプション設定」を参照してください。

ANSI_BLANKS オプション [互換性]

機能　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**

Open Client 接続と JDBC 接続の場合は **ON**

説明　ANSI_BLANKS オプションは、データベースが -b コマンド・ライン・オプションで作成されないかぎり効力を持ちません。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 では、ANSI_BLANKS が ON の場合にデータ型 DT_STRING の値を設定するときは、**sqllen** フィールドに末尾の null 文字用のスペースを含めた値の長さを設定します。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 の設定は、接続の間だけ有効です。接続が確立された後でこの設定を変更しても、その接続には影響しません。

ANSI_CLOSE_CURSORS_ON_ROLLBACK オプション [互換性]

機能 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**

説明 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、ロールバックした際、Adaptive Server 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。

ANSI_INTEGER_OVERFLOW オプション [互換性]

機能 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　　**OFF**

説明　　ISO SQL/92 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。Adaptive Server Anywhere の動作は、以前とは異なります。ANSI_INTEGER_OVERFLOW オプションを **OFF** に設定して、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]

機能　　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**

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。

デフォルト　　**ON**

説明　　ANSI_PERMISSIONS を ON にすると、DELETE 文と UPDATE 文に対する SQL/92 パーミッション要求がチェックされます。Adaptive Server Enterprise 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は UPDATE パーミッション。WHERE 句に示されるすべてのカラムでは SELECT。set 句の右側にあるすべてのカラムでは SELECT パーミッション。

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。
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 句に示されるすべてのカラムでは 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_UPDATE_CONSTRAINTS オプション [互換性]

機能 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 新しいデータベースでは **CURSORS**

バージョン 7.0 より前に作成されたデータベースでは **OFF**

説明 Adaptive Server Anywhere では、ANSI SQL 規格では許可されていない言語拡張機能がいくつか含まれており、更新を行えます。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/92 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。

このオプションが STRICT に設定されると、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 文では許可されていません。

オプションが CURSORS に設定された場合は、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/92 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　『ASA SQL リファレンス・マニュアル』> 「UPDATE 文」

ANSINULL オプション [互換性]

機能　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**

説明　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 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、次の 4 つの条件に対して Adaptive Server Anywhere は 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 **ON** の場合、少なくとも NULL 値が 1 つ含まれている式の集合関数の評価では、COUNT(*) を除いて、「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場合があります。ANSINULL OFF の場合、この警告は表示されません。

制限事項

- ANSINULL を **OFF** に設定する場合、SELECT、UPDATEDELETE、INSERT 文の WHERE、HAVING、または ON 述部のみに影響します。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** の場合の、NULL パターン文字列を持つ LIKE 述部の動作を変更しました。Adaptive Server Anywhere では、LIKE 述部は ANSINULL の設定に影響を受けないままになっています。

AUDITING オプション [データベース]

機能 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**

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。

デフォルト **OFF**

説明 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。

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を実行したい情報のタイプを指定してください。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 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 ON を設定して監査オプションを指定しない場合、あらゆるタイプの監査情報が記録されます。または、監査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することができます。

参照　『ASA SQL リファレンス・マニュアル』> 「sa_enable_auditing_type」

『ASA SQL リファレンス・マニュアル』> 「sa_disable_auditing_type」

例

- 監査をオンにします。

```
SET OPTION PUBLIC.AUDITING = 'ON'
```

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 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。

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。

AUTOMATIC_TIMESTAMP オプション [互換性]

機能 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**

説明 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。デフォルトは OFF です。

詳細については、『ASA SQL ユーザーズ・ガイド』> 「Transact-SQL との互換性を維持するためのオプション設定」を参照してください。

BACKGROUND_PRIORITY オプション [データベース]

機能 現在の接続以外の接続においてのパフォーマンスへの影響を制限します。

指定可能な値 **ON**、**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。

デフォルト　**OFF**

説明　**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。

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。

BLOB_THRESHOLD オプション [レプリケーション]

機能　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* (キロバイト)

デフォルト　**256**

説明　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。

BLOB_THRESHOLD をリモート Adaptive Server Anywhere データベースで高い値に設定すると、BLOB は細分化されず、操作は Message Agent によって Adaptive Server Enterprise に適用できます。各 SQL 文はメッセージ内に入れる必要があるので、これは小さな BLOB のレプリケーションにのみ許可されます。

BLOCKING オプション [データベース]

機能 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**

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。

デフォルト **ON**

説明 BLOCKING を ON にしておくと、あるトランザクションが既存のロックと競合するロックを取得しようとした場合、競合するすべてのロックが解放されるまで、その処理は待機します。解放されると、書き込みが実行されます。BLOCKING が OFF の場合は、競合するロックを取得しようとしたトランザクションにはエラーが返されます。

詳細については、『ASA SQL ユーザーズ・ガイド』> 「2 フェーズ・ロック」を参照してください。

BLOCKING_TIMEOUT オプション [データベース]

機能 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)

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。

デフォルト **0**

説明

BLOCKING が ON の場合、ロックを取得しようとするトランザクションは、既存のロックと競合すると、BLOCKING_TIMEOUT ミリ秒の間、競合するロックが解放されるのを待機します。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]」828 ページ

CHAINED オプション [互換性]

機能

BEGIN TRANSACTION 文がないときにトランザクション・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 JDBC 接続の場合は **OFF**

説明

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、各文は、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、個々にコミットされます。連鎖モード (CHAINED = ON) では、データ検索か修正文の前に暗黙的にトランザクションが開始されます。

CHECKPOINT_TIME オプション [データベース]

機能

データベース・サーバがチェックポイントを実行しないでいる最大分数を設定します。

指定可能な値

整数

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。

デフォルト

60

説明

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。

チェックポイントについては、「チェックポイントとチェックポイント・ログ」514 ページを参照してください。

参照

「RECOVERY_TIME オプション [データベース]」890 ページ

CIS_OPTION オプション [データベース]

機能

リモート・データ・アクセス用のデバッグ情報がサーバ・ウィンドウに表示されるかどうかを制御します。

指定可能な値

0, **7**

スコープ

個々の接続または PUBLIC グループに設定できます。

デフォルト

0

説明

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・ウィンドウで表示するか否かを制御します。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・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・ウィンドウには表示されません。

リモート追跡を有効にすると、データベース・サーバ・ウィンドウに追跡情報が表示されます。データベース・サーバの起動時に - o *filename* サーバ・オプションを指定すると、この出力をファイルに記録することができます。

-o サーバ・オプションの詳細については、『ASA データベース管理ガイド』> 「-o サーバ・オプション」を参照してください。

CIS_ROWSET_SIZE オプション [データベース]

機能　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*

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。

デフォルト　**50**

説明　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]

機能　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**

説明　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_AT_ROLLBACK オプションの設定内容で制御されます。

CLOSE_ON_ENDTRANS が **OFF** に設定されている場合は ANSI_CLOSE_CURSORS_AT_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。

これを **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。

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 SQL 文の終わりを示すコマンド・デリミタを設定できます。コマンド・デリミタは、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。このコマンド・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 (デフォルト値) 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」698 ページ

例

```
SET OPTION COMMAND_DELIMITER='~'

SET TEMPORARY OPTION command_delimiter = 'select';
 message 'hello' select
```

Interactive SQL の -d コマンド・ライン・オプションを使用しても、コマンド・デリミタを設定できます。この場合、*.sql* ファイルに SET TEMPORARY 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。

COMPRESSION オプション [レプリケーション]

機能　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　*整数* (-1 ～ 9)

デフォルト　**6**

説明　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。SQL Remote の以前のバージョンの Message Agents (dbremote と ssremote の両方) では、バージョン 6 フォーマットで送信されたメッセージを読み込めません。使用しているシステムのすべての Message Agent をバージョン 6 にアップグレードするまでは、COMPRESSION 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。

CONTINUE_AFTER_RAISERROR オプション [互換性]

機能　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　バージョン 5.x を使用して作成されたデータベースでは **OFF**

バージョン 6 以降で作成されたデータベースには **ON**

説明

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、プロシージャかトリガの実行は、RAISERROR 文を見つけるたびに停止します。

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らなくなります。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ために使用します。ただし、使用されるのは 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*だけ*です。ON_TSQL_ERROR オプションに STOP または CONTINUE を設定した場合は、ON_TSQL_ERROR 設定が CONTINUE_AFTER_RAISERROR 設定に優先します。

参照

「ON_TSQL_ERROR オプション [互換性]」875 ページ

CONVERSION_ERROR オプション [互換性]

機能

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

説明　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のデータ型変換障害が、エラー (CONVERSION_ERROR が **ON**) と警告 (CONVERSION_ERROR が **OFF**) のどちらとしてデータベースにレポートされるかを制御します。

CONVERSION_ERROR を **ON** に設定すると、SQLE_CONVERSION_ERROR エラーが出されます。オプションを **OFF** に設定すると、警告 SQLE_CANNOT_CONVERT が出力されます。

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]

機能　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)

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。

デフォルト　**250**

説明　このオプションは、COOPERATIVE_COMMITS が **ON** に設定されているとき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。

COOPERATIVE_COMMITS オプション [データベース]

機能　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　または　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。

デフォルト **ON**

説明 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) に設定した場合と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 の数が減ってデータベース・サーバ・スループットが全体的に増加しますが、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、COOPERATIVE_COMMIT_TIMEOUT 間隔をページに書き込まないで渡すと、アプリケーションが再開して (コミットが作動したように)、残りの間隔 (DELAYED_COMMIT_TIMEOUT - COOPERATIVE_COMMIT_TIMEOUT) は DELAYED_COMMIT 間隔として使用されます。その後、ページが完全でなくても書き込まれます。

DATE_FORMAT オプション [互換性]

機能 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*

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。

デフォルト *'YYYY-MM-DD'*。これは、ISO 日付フォーマット仕様に対応します。

説明

フォーマットは次の記号を組み合わせた文字列です。

記号	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
d	曜日を示す 1 桁の数字 (0 = 日曜、6 = 土曜)
dd	2 桁の日
ddd[*d*…]	曜日を略した文字
hh	2 桁の時間
nn	2 桁の分
ss[.*ss*..]	秒数とコンマ以下の秒数
aa	AM か PM (12 時間表記)
pp	必要であれば PM (12 時間表記)
jjj	通し日数 (1 ～ 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、フォーマットで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、フォーマットで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 Adaptive Server 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)、ゼロ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。

例

- 次の表は、2001 年 5 月 21 日 (木) に実行された次の文からの出力と、DATE_FORMAT 設定を示します。

```
SELECT CURRENT DATE
```

DATE_FORMAT	SELECT CURRENT DATE
yyyylmmlddlddd	2001/05/21/thu
jjj	141
mmm yyyy	May 1998
mm-yyyy	05-1998

DATE_ORDER オプション [互換性]

機能 日付フォーマットの解釈を制御します。

指定可能な値 **'MDY'**、**'YMD'**、**'DMY'**

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。

デフォルト　**YMD**。これは、ISO 日付フォーマット仕様に対応します。

Open Client 接続と JDBC 接続の場合、デフォルトでは **MDY** に設定されます。

説明　データベース・オプション DATE_ORDER は、10/11/12 が Oct 11 1912、Nov 12 1910、Nov 10 1912 のいずれを意味するかを指定するために使用します。このオプションには、'MDY'、'YMD'、または 'DMY' の値が使用できます。

DEBUG_MESSAGES オプション [データベース]

機能　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**

説明　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・プロシージャおよびトリガ内のデバッグ・メッセージの動作を制御できます。デフォルトで、このオプションは OFF に設定されていて、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 に設定する場合、すべてのストアド・プロシージャおよびトリガでデバッグ・メッセージを有効にします。

注意
DEBUG_MESSAGES オプションが OFF に設定されている場合、DEBUG ONLY メッセージは容量が小さいため、通常運用システムのストアド・プロシージャに残されます。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
- 『ASA SQL リファレンス・マニュアル』>「MESSAGE 文」

DEDICATED_TASK オプション [データベース]

機能 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**

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。

デフォルト **OFF**

説明 DEDICATED_TASK 接続オプションが ON に設定されている場合、要求処理タスクはもっぱらその接続の要求のみを処理します。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限りデータベース・サーバのステータスに関する情報を集められます。

DEFAULT_ISQL_ENCODING オプション [Interactive SQL]

機能 READ、INPUT、および 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*

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。

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。

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 Windows マシンでは、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、OUTPUT、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 が開始されたときに -codepage コマンド・ライン・オプションで指定されたコード・ページ
- Interactive SQL が動作しているコンピュータ用のデフォルトのコード・ページ

サポートされるコード・ページの完全なリストについては、「サポートされているコード・ページ」473 ページを参照してください。

参照

- 『ASA SQL リファレンス・マニュアル』>「READ 文 [Interactive SQL]」
- 『ASA SQL リファレンス・マニュアル』>「INPUT 文 [Interactive SQL]」
- 『ASA SQL リファレンス・マニュアル』>「OUTPUT 文 [Interactive SQL]」
- 「文字セットの問題」422 ページ

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 TEMPORARY OPTION DEFAULT_ISQL_ENCODING = 'UTF-16'
```

DEFAULT_TIMESTAMP_INCREMENT オプション [データベース]

機能

TIMESTAMP のデータ型を持つカラムに追加する、マイクロ秒の数値を指定して、カラム中の値をユニークにします。

指定可能な値

整数 (1 ～ 60 000 000)

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。

デフォルト

1

説明

Adaptive Server Anywhere では、TIMESTAMP 値は小数第 6 位の精度なので、デフォルトでは 1 マイクロ秒 (0.000001 秒) が 2 つの同一の TIMESTAMP 値間の区別を付けるために追加されます。

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。互換性を維持するために、TRUNCATE_TIMESTAMP_VALUES オプションを ON に設定してから、Adaptive Server Anywhere が格納する小数点以下の桁数を指定できます。

Mobile Link 同期の場合、最初の同期の実行前にこのオプションを設定してください。

参照

「TRUNCATE_TIMESTAMP_VALUES オプション [データベース]」906 ページ

DELAYED_COMMIT_TIMEOUT オプション [データベース]

機能

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

整数 (ミリ秒)

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。

デフォルト

500

説明

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 エントリがいつディスクに書き込まれるかを管理します。DELAYED_COMMITS を **ON** に設定すると、データベース・サーバは、現在のページ内容をディスクに書き込む前に、他の接続がログのページを埋めるのを DELAYED_COMMIT_TIMEOUT オプションで指定したミリ秒間待ちます。

詳細については、「DELAYED_COMMITS オプション [データベース]」843 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]

機能　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**

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。

デフォルト　**OFF**。これは、ISO の COMMIT 動作に対応します。

説明　**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、すぐに 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せん。

このオプションが **ON** の場合、ログは、ログ・ページが一杯なときか、DELAYED_COMMIT_TIMEOUT オプション設定に応じて、どちらか早い時点で、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、セキュリティを犠牲にして早い応答時間が得られます。

COOPERATIVE_COMMITS と DELAYED_COMMITS の両方を **ON** に設定しているとき、ページが書き込まれないまま COOPERATIVE_COMMIT_TIMEOUT 間隔が経過すると、(コミットが行われたかのように) アプリケーションが再開されます。残りの間隔 (DELAYED_COMMIT_TIMEOUT - COOPERATIVE_COMMIT_TIMEOUT) は、DELAYED_COMMIT 間隔として使用され、その間隔が経過すると、ページが一杯でなくても書き込みが行われます。

DELETE_OLD_LOGS オプション [レプリケーション]

機能 トランザクション・ログのメッセージがレプリケートまたは同期されるときに、トランザクション・ログが削除されるかどうかを制御します。

指定可能な値 **ON**、**OFF**、**DELAY**

デフォルト **OFF**

説明 このオプションは、Mobile Link クライアント、SQL Remote と Adaptive Server 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が送信され、その受信が確認されたときに、古いトランザクション・ログは削除されます。**DELAY** に設定すると、実行日に作成されたことを示すファイル名を持つ古いトランザクション・ログは削除されません。

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、『ASA SQL リファレンス・マニュアル』> 「BACKUP 文」を参照してください。

DESCRIBE_JAVA_FORMAT オプション [Interactive SQL]

機能 Java オブジェクトが文字列 (表示用) またはバイナリ (ロードとアンロード用) のどちらとして解釈されるかを指定します。

Java オブジェクトがバージョン 8 以降のオプションでサポートされていないため、このオプションはバージョン 8 以前のデータベース・サーバに接続している場合のみ使用します。

指定可能な値 **varchar**、**binary**

デフォルト **varchar**

説明

varchar を設定すると、Interactive SQL は、データベースからフェッチされたデータをすべて文字列に変換します。データベース・サーバは、Java カラムで `toString()` メソッドを呼び出して、[結果] ウィンドウ枠の [結果] タブに対する出力をフォーマットします。

binary を設定すると、Interactive SQL はデータ変換を行いません。

DIVIDE_BY_ZERO_ERROR オプション [互換性]

機能

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

説明

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。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。

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。

ロギングをオンにする場合の詳細については、『ASA SQL リファレンス・マニュアル』> 「START LOGGING 文 [Interactive SQL]」を参照してください。

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確保されています。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確保されています。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]

機能 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**

説明 このオプションが **ON** に設定されている場合、Adaptive Server Anywhere は重複した相関名を外部ジョインの NULL 入力側に使用することを許可します。同じ相関名で指定された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ったジョインの Adaptive Server Anywhere による解釈を示しています。

```
( R left outer join T on (C1), T  join S on ( C2 ) )
```

ここで、C1 と C2 は探索条件です。

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。

```
ASA 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　『ASA SQL リファレンス・マニュアル』> 「REWRITE 関数 [その他]」

FIRE_TRIGGERS オプション [互換性]

機能　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**

説明　**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、FIRE_TRIGGERS 設定に関わらずすべてのトリガ起動をオフにする -gf コマンド・ライン・オプションによって上書きされます。

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て、すべて Adaptive Server 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 Adaptive Server Anywhere にデータをレプリケートするときに意味があります。

FIRST_DAY_OF_WEEK オプション [データベース]

機能　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)

説明

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

FLOAT_AS_DOUBLE オプション [互換性]

機能

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**

説明

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)、すべてのキーワード FLOAT は SQL 文内のキーワード DOUBLE と同等に解釈されます。

デフォルトでは、Adaptive Server 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 Adaptive Server 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/92 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 Adaptive Server 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]

機能　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**

説明　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　『ASA SQL ユーザーズ・ガイド』> 「FOR XML 句を使用してクエリ結果を XML として取り出す」

『ASA SQL リファレンス・マニュアル』> 「SELECT 文」

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確保されています。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]

機能
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。デフォルトのグローバル・オートインクリメントで作成されたカラムで、値の範囲の先頭を制御します。

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。

説明
デフォルトのグローバル・オートインクリメントでカラムを作成すると、カラムの値が 1 つずつ増えます。初期値は、GLOBAL_DATABASE_ID 値と分割サイズによって決まります。

パーティション・サイズの詳細については、『ASA SQL リファレンス・マニュアル』> 「CREATE TABLE 文」を参照してください。

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。

詳細については、『SQL Remote ユーザーズ・ガイド』>「ユニークなプライマリ・キーの確保」と『Mobile Link 管理ガイド』>「グローバル・オートインクリメントを使用したユニークなプライマリ・キーの管理」を参照してください。

参照

『ASA SQL リファレンス・マニュアル』>「CREATE TABLE 文」

「データベース・レベルのプロパティ」946 ページ

『SQL Remote ユーザーズ・ガイド』>「ユニークなプライマリ・キーの確保」

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。以下を参照。

デフォルト

ASCII

説明

特定のファイル・フォーマットには、カラム名と型に関する情報が含まれています。データベース・テーブルが存在していない場合、INPUT 文はこの情報を使ってデータベース・テーブルを作成します。これは、データベースにデータをロードする非常に簡単な方法です。テーブルの作成に十分な情報を持つフォーマットは、DBASEII、DBASEIII、FOXPRO、LOTUS です。

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 **¥x*DD*** 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、ファイルの中の情報に基づいて決定しようとします。テーブルがない場合は、テーブルが作成されます。
- **DBASEII** このファイルは dBASE II フォーマットです。テーブルがない場合は、テーブルが作成されます。
- **DBASEIII** このファイルは dBASE III フォーマットです。テーブルがない場合は、テーブルが作成されます。
- **EXCEL** 入力ファイルは Microsoft Excel 2.1 のフォーマットです。テーブルがない場合は、テーブルが作成されます。
- **FIXED** 入力行は固定フォーマットです。カラムの幅は、COLUMN WIDTHS 句を使って指定できます。カラムの幅を指定しない場合、ファイル内のカラム幅は、対応するデータベース・カラムの型の値に必要な文字の最大値と同じにしてください。
- **FOXPRO** ファイルは FoxPro フォーマットです。FoxPro メモ・フィールドは dBASE メモ・フィールドとは異なります。テーブルがない場合は、テーブルが作成されます。
- **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。テーブルがない場合は、テーブルが作成されます。この場合、作成されたカラムの型とサイズは正確ではありません。ファイル内の情報がカラムではなく、セルに属しているからです。

参照 『ASA SQL リファレンス・マニュアル』> 「INPUT 文 [Interactive SQL]」

INTEGRATED_SERVER_NAME オプション [データベース]

機能　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*

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。

デフォルト　NULL

説明　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・メンバーシップの確認に Adaptive Server Anywhere データベース・サーバを実行中のマシンが使用されます。

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

『ASA SQL リファレンス・マニュアル』> 「GRANT 文」

例　次の例では、グループ・メンバシップがマシン・サーバ 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]

機能　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**

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。

デフォルト　**0**

Open Client 接続と JDBC 接続の場合は **1**

説明

このオプションは次のとおり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・リードを防ぎ、繰り返し可能読み出しを保証します。幻ローを許可します。
- **3** 直列化可能。ダーティ・リードを防ぎ、繰り返し可能読み出しを保証し、幻ローを許可しません。

詳細については、『ASA SQL ユーザーズ・ガイド』> 「独立性レベルと一貫性」を参照してください。

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。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] ウィンドウ枠に表示されます。オプションを **OFF** に設定すると、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。

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。

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 TEMPORARY 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'

  # はエスケープ文字、*x0A* は "¥n" を 16 進に変換したものです。

  先頭と最後の文字 (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。

例

- *c:¥employee.txt* にエクスポートするデータで、フィールド・セパレータとしてコロンを指定します。

  ```
  SET TEMPORARY OPTION ISQL_FIELD_SEPARATOR=':';
   SELECT emp_lname, emp_fname FROM employee WHERE
  emp_id < 150;
   OUTPUT TO c:¥employee.txt FORMAT ASCII
  ```

  このコードを実行すると、次のデータが *employee.txt* に書き込まれます。

  'Whitney':'Fran'

  'Cobb':'Matthew'

  'Chin':'Philip'

  'Jordan':'Julie'

  先頭と最後の文字 (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_LOG オプション [Interactive SQL]

機能 ロギング動作を制御します。

指定可能な値 *文字列* (ファイル名)

デフォルト　*空の文字列*

説明　ISQL_LOG を空でない文字列に設定すると、指定されたファイルの終わりにすべての Interactive SQL 文が追加されます。ISQL_LOG を空の文字列に設定した場合は、Interactive SQL 文のログは取られません。

対象は個別セッションのみ
このオプションは、個別の Interactive SQL セッションでのみログを取ります。

すべてのユーザによるデータベースへの変更をすべて記録するトランザクション・ログについては、「バックアップとデータ・リカバリ」 483 ページを参照してください。

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] タブと [Ultra Light 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**、**GRAPHICALWITHSTATISTICS**、**NONE**

デフォルト　**SHORT**

説明　このオプションを使用して、SELECT、INSERT、UPDATE、および 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　クエリのツリー状の図が表示されます。このプラン図内の 1 つのノードをクリックすると、クエリのその部分の詳細を表示できます。

GRAPHICALWITHSTATISTICS 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。Ultra Light プランでは、この値は GRAPHICAL 値と同様に処理されます。

NONE オプティマイザ情報にアクセスできないことを意味します。このパラメータは推奨されなくなりました。

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。

ISQL_PRINT_RESULT_SET オプション [Interactive SQL]

機能 このオプションは、*.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) にのみ有効です。

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、*.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。このオプションは、(ユーザごとではなく) マシンごとに保存され、大文字と小文字は区別されません。

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] ー [オプション] を選択してから、[結果] ページの [コンソール・モード] ボックスで適切な処置を選択します。Interactive SQL のすべてのセッションに対してこのオプションを設定する場合は、[設定] をクリックします。そうしないと、Interactive SQL を終了するときに変更内容が破棄されます。

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。

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 TEMPORARY OPTION ISQL_QUOTE='"';
 SELECT emp_lname, emp_fname FROM employee WHERE
emp_id < 150;
 OUTPUT TO c:¥employee.txt FORMAT ASCII
```

このコードを実行すると、次のデータが *employee.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。

JAVA_HEAP_SIZE オプション [データベース]

機能 接続で Java アプリケーションが使用するメモリを制限します。

指定可能な値 *整数*

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。

デフォルト **1 000 000**

説明 このオプションは、接続単位で Java アプリケーションに割り付けられるメモリの最大サイズ (単位 : バイト) を設定します。通常、接続ごとのメモリ割り付けは、割り付けられた Java 変数のユーザの作業セットと Java アプリケーション・スタック領域から構成されます。

Java アプリケーションの実行中、接続ごとの割り付けはデータベース・サーバの固定キャッシュを使用するため、暴走して制御できなくなった Java アプリケーションがメモリを使いすぎないようにすることが重要です。

JAVA_INPUT_OUTPUT オプション [データベース]

機能 データベース内の Java からのファイル・アクセスを有効にします。

指定可能な値 **ON、OFF**

Windows NT のみでサポートされています。

スコープ DBA 権限が必要です。

デフォルト　　**OFF**

説明　　デフォルトでは、**java.io** パッケージは一部分だけがサポートされています。特に、ファイル・アクセスを処理するクラスは無効になっています。JAVA_INPUT_OUTPUT が **ON** に設定されている場合は、**java.io** の Java ファイル・アクセス・クラスが有効になります。

参照　　『ASA プログラミング・ガイド』> 「Java のセキュリティ管理」

JAVA_NAMESPACE_SIZE オプション [データベース]

機能　　接続で Java アプリケーションが使用するメモリを制限します。

指定可能な値　　*整数*

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。

デフォルト　　**4 000 000**

説明　　このオプションは、データベース単位で Java アプリケーションに割り付けられるメモリの最大サイズ (単位：バイト) を設定します。

データベースごとのメモリ割り付けは、Java クラス定義を含みます。クラス定義は読み取り専用であるため、接続間で共有されます。したがって、その割り付けには固定キャッシュを使用することになり、このオプションを使って割り付けサイズの制限を設定します。

LOG_DEADLOCKS オプション [データベース]

機能　　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**

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。

デフォルト　　**ON**

説明

このオプションが ON に設定されている場合、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。このオプションが OFF に設定されていると、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

『ASA SQL リファレンス・マニュアル』> 「sa_report_deadlocks システム・プロシージャ」

『ASA SQL ユーザーズ・ガイド』> 「ブロックされているユーザの判別」

LOGIN_MODE オプション [データベース]

機能

データベースの統合化ログインの使用を制御します。

指定可能な値

Standard、**Mixed**、**Integrated**

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。

デフォルト

Standard

説明

このオプションは、統合化ログインが許可されるかどうかを指定します。次の値を指定できます (大文字と小文字は区別されません)。

- **Standard** これはデフォルトの設定であり、統合化ログインは使用できません。統合化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。
- **Mixed** この設定では、統合化ログインと標準ログインの両方を使用できます。

- **Integrated**　データベースへのすべてのログインを、統合化ログインを使用して行います。

警告
LOGIN_MODE データベース・オプションを **Integrated** に設定すると、接続できるのは、統合化ログイン・マッピングを付与されている Windows 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 (完全な管理権) を持つユーザです。

統合化ログインの詳細については、「統合化ログインの使用方法」97 ページを参照してください。

LOGIN_PROCEDURE オプション [データベース]

機能　起動時の接続互換性オプションを設定するログイン・プロシージャです。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*

スコープ　DBA 権限が必要です。

デフォルト　**sp_login_environment**

説明　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。

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。sp_login_environment または sp_tsql_environment は編集しないでください。

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については、『ASA SQL リファレンス・マニュアル』> 「RAISERROR 文 [T-SQL]」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
```

MAX_CURSOR_COUNT オプション [データベース]

機能　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・ガバナーです。

指定可能な値　*整数*

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。

デフォルト　**50**

説明　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_HASH_SIZE オプション [データベース] (旧式)

機能　新しいインデックスのデフォルトの最大ハッシュ・サイズを指定します。

指定可能な値　*整数* (2 ～ 64)

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。

デフォルト　**10**

説明　このオプションでは、インデックスのデフォルト・ハッシュ・サイズを制御します。このオプションは推奨されなくなりました。

参照　『ASA SQL リファレンス・マニュアル』>「CREATE INDEX 文」

『ASA SQL リファレンス・マニュアル』>「CREATE TABLE 文」

MAX_PLANS_CACHED オプション [データベース]

機能　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*

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。

デフォルト　**20**

説明　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ほう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

『ASA SQL ユーザーズ・ガイド』> 「アクセス・プランのキャッシュ」

「接続レベルのプロパティ」923 ページ

MAX_RECURSIVE_ITERATIONS オプション [データベース]

機能

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

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。

デフォルト

100

説明

終了の失敗が指定した回数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。無限反復が検出される前に消費される時間とリソースの総量を制限するためにこのオプションを設定しますが、意図した通りに再帰共通テーブル式も動作できます。

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

『ASA SQL ユーザーズ・ガイド』> 「共通テーブル式」

MAX_STATEMENT_COUNT オプション [データベース]

機能

準備文のうち、1 つの接続において一度に使用できる最大数を制限するリソース・ガバナーです。

指定可能な値　0 以上の整数

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。

デフォルト　**50**

説明　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準備文の数を、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_WORK_TABLE_HASH_SIZE オプション [データベース] (旧式)

機能　内部テンポラリ・テーブルに対するクエリの最適化に使用する最大ハッシュ・サイズを指定します。

指定可能な値　*整数* (2 ～ 64)

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。

デフォルト　**20**

説明　クエリ・オプティマイザが内部テンポラリ・テーブル内のデータ分散に基づいてテーブルのハッシュ・サイズを割り当てるため、通常はこのオプションを使用する必要はありません。このオプションはオプティマイザの動作を変更して、オプティマイザが使用できる最大ハッシュ・サイズを 20 より大きくするか、逆にさらに少なく制限します。このオプションは推奨されなくなりました。

参照 『ASA SQL ユーザーズ・ガイド』> 「クエリ処理中のワーク・テーブルの使用」

MIN_PASSWORD_LENGTH オプション [データベース]

機能 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* (0 以上)

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。

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。

デフォルト **0** 文字

説明 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することができます。既存のパスワードは影響を受けません。パスワードの長さは最大で 255 バイトです。

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]

機能 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 ～ 100)

デフォルト バージョン 6 以降で作成されたデータベースには **50**

バージョン 5.5 以前で作成されたデータベースには **0**

説明 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 Adaptive Server 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]

機能　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

説明　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、リリース 4.0d 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 『ASA SQL リファレンス・マニュアル』> 「キーワード」

NULLS オプション [Interactive SQL]

機能 データベース内の NULL 値の表示方法を指定します。

指定可能な値 *文字列*

デフォルト **(NULL)**

説明 好みに応じて設定します。

ODBC_DESCRIBE_BINARY_AS_VARBINARY [データベース]

機能 Adaptive Server 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**

説明 このオプションを使用すると、すべての BINARY および 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、Adaptive Server 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 および 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変数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

- 『ASA SQL リファレンス・マニュアル』>「BINARY データ型 [バイナリ]」
- 『ASA SQL リファレンス・マニュアル』>「VARBINARY データ型 [バイナリ]」

ODBC_DISTINGUISH_CHAR_AND_VARCHAR オプション [データベース]

機能

Adaptive Server 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

説明

接続を開くと、Adaptive Server Anywhere ODBC ドライバが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。

参照

- 『ASA SQL リファレンス・マニュアル』>「CHAR データ型 [文字]」
- 『ASA SQL リファレンス・マニュアル』>「CHARACTER VARYING (VARCHAR) データ型 [文字]」

ON_CHARSET_CONVERSION_FAILURE オプション [データベース]

機能 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 *文字列*。指定可能な値は以下を参照してください。

デフォルト **IGNORE**

説明 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。

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。

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してください。

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] 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] 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 ファイルを実行している場合、STOP および EXIT は同じです。

ON_TSQL_ERROR オプション [互換性]

機能　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してください。

デフォルト　**CONDITIONAL**

説明　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 用の CONDITIONAL と CONTINUE 設定が、Adaptive Server Enterprise との互換性を維持するために使用されます。CONTINUE が Adaptive Server Enterprise 動作を最も厳密にシミュレー

トしています。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、エラーのレポートがより迅速なため CONDITIONAL 設定をおすすめします。

このオプションに STOP または CONTINUE を設定したときは、その設定が CONTINUE_AFTER_RAISERROR オプションの設定に優先します。ただし、このオプションに CONDITIONAL (デフォルト) を設定したときは、RAISERROR 文に続く動作は CONTINUE_AFTER_RAISERROR オプションが決定します。

参照

『ASA SQL リファレンス・マニュアル』> 「CREATE PROCEDURE 文」

『ASA SQL リファレンス・マニュアル』> 「CREATE PROCEDURE 文 [T-SQL]」

『ASA SQL ユーザーズ・ガイド』> 「Transact-SQL のプロシージャ言語の概要」

「CONTINUE_AFTER_RAISERROR オプション [互換性]」 833 ページ

OPTIMISTIC_WAIT_FOR_COMMIT オプション [互換性]

機能

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

説明

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** の場合に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、5.x アプリケーションを 8.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限り) 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、5.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)。

OPTIMIZATION_GOAL オプション [データベース]

機能　クエリ処理に対し、最初のローを迅速に返すようにするか、完全な結果セットを返す負荷を最小限に抑えるようにするかを決定し、クエリを最適化する方法を指定します。

指定可能な値　**first-row**、**all-rows**

デフォルト　**all-rows**

説明　OPTIMIZATION_GOAL オプションは、Adaptive Server 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、Adaptive Server 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 に **all-rows** を設定するのが

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、Adaptive Server 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 Adaptive Server 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 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、『ASA SQL リファレンス・マニュアル』> 「FROM 句」を参照してください。

OPTIMIZATION_LEVEL オプション [データベース]

機能 Adaptive Server 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**

説明 OPTIMIZATION_LEVEL オプションは、Adaptive Server 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。

オプションが 0 に設定された場合、Adaptive Server 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 および 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」923 ページを参照してください。

OPTIMIZATION_WORKLOAD オプション [データベース]

機能

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**

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。

デフォルト **Mixed**

説明 OPTIMIZATION_WORKLOAD オプションは、Adaptive Server 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、Adaptive Server 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、Adaptive Server 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。

参照 『ASA SQL ユーザーズ・ガイド』>「クラスタード・ハッシュ GROUP BY アルゴリズム」

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してください。

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 句で 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 (FoxPro メモ・フィールドは dBASE メモ・フィールドとは異なります)。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** この出力は HTML (Hyper Text Markup Language) 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 でコード化され、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 ブロック内にコード化されます。

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 *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。

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、および SQL の出力フォーマットだけに影響します。

PERCENT_AS_COMMENT オプション [互換性]

機能　　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**

説明　　% 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/92 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、Adaptive Server 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。

既存のプロシージャはオプションを変更する前に再作成が必要
% スタイルのコメントがある既存のプロシージャは、オプション設定を変更する前に再作成してください。そうしないと、プロシージャをロードできません。

PINNED_CURSOR_PERCENT_OF_CACHE オプション [データベース]

機能 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 ～ 100)

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。

デフォルト **10**

説明 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 ～ 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。

PRECISION オプション [データベース]

機能 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* (0 ～ 127)

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。

デフォルト　**30**

説明　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、小数点以下の最大桁数を指定します。

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]

機能　PREFETCH 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。

指定可能な値　**OFF**、**CONDITIONAL**、**ALWAYS**

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。

デフォルト　**CONDITIONAL**

説明　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限り、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 の値は、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 の設定は、Open Client 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。

このオプションは以前は値の ON を受け入れていました。この値は現在は CONDITIONAL のエイリアスです。

参照

- 『ASA プログラミング・ガイド』> 「ローのプリフェッチ」
- 「DisableMultiRowFetch 接続パラメータ [DMRF]」255 ページ

PRESERVE_SOURCE_FORMAT オプション [データベース]

機能

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、SYSTABLE、SYSPROCEDURE、SYSTRIGGER、SYSEVENT 内のカラム source に保存されます。

指定可能な値　**ON、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。

デフォルト　**ON**

説明　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・テーブルの source 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]

機能　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**

説明　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。このオプションは、特にレプリケーション環境と同期環境でデータの整合性を保証するときに役立ちます。

QUALIFY_OWNERS オプション [レプリケーション]

機能 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 Adaptive Server Anywhere でのデフォルトは **ON** です。

Adaptive Server Enterprise でのデフォルトは **OFF** です。

説明 Adaptive Server Enterprise 設定で所有者を修飾する必要はありません。一般的に、オブジェクトは **dbo** によって所有されているためです。Adaptive Server 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。

QUERY_PLAN_ON_OPEN オプション [互換性]

機能 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**

説明 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。デフォルトでは、**OFF** に設定されています。

QUOTE_ALL_IDENTIFIERS オプション [レプリケーション]

機能 SQL Remote によってレプリケートされた SQL 文で、識別子を引用符で囲んで使用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　**OFF**

説明　このオプションが **OFF** の場合、dbremote は Adaptive Server Anywhere によって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、*ssremote* は識別子を引用符で囲みません。

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。

QUOTED_IDENTIFIER オプション [互換性]

機能　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 JDBC 接続の場合は **OFF**

説明　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。

Interactive SQL では、データベースに接続するときに、次の文を自動的に実行します。

```
SET TEMPORARY OPTION QUOTED_IDENTIFIER = ON
```

この TEMPORARY オプションは、現在の接続に対してのみ適用され、その接続が終了するまで持続します。

詳細については、『ASA SQL ユーザーズ・ガイド』> 「Transact-SQL との互換性を維持するためのオプション設定」を参照してください。

READ_PAST_DELETED オプション [データベース]

機能 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**

説明 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]

機能 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)

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。

デフォルト **2**

説明 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。

Adaptive Server 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。そのため、リカバリ時間は正確ではありません。

詳細については、「自動リカバリ処理」517 ページを参照してください。

REMOTE_IDLE_TIMEOUT オプション [データベース]

機能　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**

説明　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 [レプリケーション]

REPLICATION_ERROR オプション [レプリケーション]

機能　SQL Remote に対して、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*

説明　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。

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。

REPLICATION_ERROR_PIECE オプション [レプリケーション]

機能

SQL Remote で REPLICATION_ERROR_PIECE オプションを REPLICATION_ERROR オプションとともに使用すると、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。

これは、LONG VARCHAR をサポートしない古いバージョンの Adaptive Server Enterprise で作成されたデータベースにとって、最も便利な互換性機能です。

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

説明

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 が最初の部分とともに呼び出され、REPLICATION_ERROR_PIECE が残りの部分とともに繰り返し呼び出されます。

RETURN_DATE_TIME_AS_STRING オプション [データベース]

機能

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**

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。

デフォルト　OFF

説明

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]」836 ページ

「TIME_FORMAT オプション [互換性]」902 ページ

「TIMESTAMP_FORMAT オプション [互換性]」903 ページ

RETURN_JAVA_AS_STRING オプション [データベース]

機能　クエリが実行された場合、クライアント・アプリケーションに Java オブジェクトが渡される方法を制御します。

指定可能な値　ON、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。

デフォルト　OFF

説明

このオプションでは、Open Client または jConnect を介して接続しているアプリケーションに対して、Java オブジェクトがどのように返されるかを指定します。

デフォルトでは、Java オブジェクトはオブジェクトの Sun Microsystems の直列化として TDS を介して返されます。クライアント、すなわちオブジェクトの直列化の受信側が、オブジェクトを非直列化し、インスタンスにする必要があります。

RETURN_JAVA_AS_STRING が **ON** に設定されている場合は、オブジェクトはまず **toString()** メソッドによって **java.lang.String** のインスタンスに変換され、その後、その String オブジェクトがクライアントに返されます。

RI_TRIGGER_TIME オプション [互換性]

機能 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**

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。

デフォルト **AFTER**

説明 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 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]

関数 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**

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。

デフォルト **ON**

説明 このオプションが ON に設定されている場合、デッドロックが発生すると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。このオプションが OFF に設定されている場合、Adaptive Server 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、『ASA SQL ユーザーズ・ガイド』>「デッドロック」を参照してください。

ROW_COUNTS オプション [データベース]

機能　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**

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。

デフォルト　**OFF**

説明　このオプションを **OFF** に設定すると、通常、ローの数は予測値のみになります。**ON** に設定すると、ローの数を正確にカウントします。

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場合があります。通常、この設定によって、Adaptive Server Anywhere がクエリを 2 度実行するので、実行時間が倍増します。

SCALE オプション [データベース]

機能　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 ～ 127)

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。

デフォルト 6

説明 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。

詳細については、「PRECISION オプション [データベース]」884 ページ を参照してください。

SORT_COLLATION オプション [データベース]

機能 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* のいずれか

デフォルト **INTERNAL**

説明 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 『ASA SQL リファレンス・マニュアル』> 「SORTKEY 関数 [文字列]」

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したときに、次のような変形が行われます。

```
SELECT name, id
 FROM product
 ORDER BY name, id
```

および

```
SELECT name, id
 FROM product
 ORDER BY 1, 2
```

が次のようになります。

```
SELECT name, id
 FROM product
 ORDER BY SORTKEY(name, 'binary'), id
```

SQL_FLAGGER_ERROR_LEVEL オプション [互換性]

機能　指定された SQL/92 セットの一部ではない SQL への応答を制御します。

指定可能な値　**E、I、F、W**

デフォルト　**W**

説明　このオプションは、指定された SQL/92 セットの一部ではない SQL をエラーとして通知します。

使用できる値は次のとおりです。

- **E**　初級レベル SQL/92 構文でない構文を通知します。
- **I**　中級レベル SQL/92 構文でない構文を通知します。
- **F**　上級 SQL/92 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。

SQL_FLAGGER_WARNING_LEVEL オプション [互換性]

機能　指定された SQL/92 セットの一部ではない SQL への応答を制御します。

指定可能な値　**E、I、F、W**

デフォルト　**W**

説明　このオプションは、指定された SQL/92 セットの一部ではない SQL を警告として通知します。

使用できる値は次のとおりです。

- **E** 初級レベル SQL/92 構文でない構文を通知します。
- **I** 中級レベル SQL/92 構文でない構文を通知します。
- **F** 上級 SQL/92 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。

STATISTICS オプション [Interactive SQL]

機能 実行時間を有効にするかどうかを制御します。

指定可能な値 *負でない任意の 整数*

デフォルト **7**

説明 このオプションは ISQL_PLAN オプションの古いバージョンで、`dbisqlc` でのみ使用されます。

STATISTICS オプションを 0 に設定した場合は、SQL 文を実行しても [メッセージ] ウィンドウ枠には何の情報も表示されません。このオプションに 0 以外の数字を設定すると、文を実行した後に、実行時間が [メッセージ] ウィンドウ枠に表示されます。ここで指定した数値は、[メッセージ] ウィンドウ枠のデフォルトの高さ (行数) としても使用されます。

参照 「ISQL_COMMAND_TIMING オプション [Interactive SQL]」854 ページ

「ISQL_PLAN オプション [Interactive SQL]」857 ページ

STRING_RTRUNCATION オプション [互換性]

機能 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　　**OFF**

説明　　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/92 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。

SUBSCRIBE_BY_REMOTE オプション [レプリケーション]

機能　　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**

説明　　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。

SUBSUME_ROW_LOCKS オプション [データベース]

機能　　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**

説明　　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・ロックを獲得しなくなります。

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 (> ~2-4m 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]

機能 TDS デバッグ情報がサーバ・ウィンドウに表示され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**

説明 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・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・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・ウィンドウに表示されます。

TDS_EMPTY_STRING_IS_NULL オプション [データベース]

機能 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**

説明　　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。このオプションに **ON** を設定すると、空の文字列は、TDS 接続では NULL 文字列で返されます。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]

機能　　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**

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。

デフォルト　　**OFF**

説明　　TEMP_SPACE_LIMIT_CHECK が OFF (デフォルト) に設定されている場合、データベース・サーバは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ません。接続で指定値以上のテンポラリ領域が要求された場合、このオプションが OFF に設定されていると致命的エラーが発生することがあります。このオプションが ON に設定されている場合、接続で指定値以上のテンポラリ・ファイル領域が要求されると、要求は失敗し、エラー SQLSTATE_TEMP_SPACE_LIMIT が返されます。

接続用のテンポラリ・ファイルの領域サイズは、テンポラリ・ファイルの最大サイズと、アクティブなデータベースの接続数という 2 つの要因によって決定されます。テンポラリ・ファイルの最大サイズは、ファイルの現在のサイズと、そのファイルが含まれているパーティションの空き領域の合計です。制限チェックが有効になっていると、テンポラリ・ファイルが最大サイズの 80% 以上になり、*さらに*接続で追加のテンポラリ・ファイル領域が要求されると、その接続が指定の領域サイズを越えていないかが確認されます。このチェックが行わ

れると、テンポラリ・ファイル最大領域をアクティブ接続数で割った数値で表した領域サイズよりも大きいサイズを使用している接続はすべて失敗します。

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 システム・プロシージャを使用すると取得できます。

参照 『ASA SQL リファレンス・マニュアル』> 「sa_disk_free_space システム・プロシージャ」

TIME_FORMAT オプション [互換性]

機能 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)

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。

デフォルト *HH*:*NN*:*SS*.*SSS*

説明 フォーマットは次の記号を組み合わせた文字列です。

- **hh** 2 桁の時間 (24 時間表記)
- **nn** 2 桁の分
- **mm** コロンの後の場合は、2 桁の分 (hh:mm など)
- **ss[.s…]** 2 桁の秒と、オプションで小数

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。

TIME_ZONE_ADJUSTMENT オプション [データベース]

機能　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* (たとえば 300)、*負の整数* (全体を引用符で囲む) (たとえば '-300')、*文字列* (先頭に + または - が付いた時刻 (時間と分) 全体を引用符で囲む) (たとえば '+5:00' や '-5:00')

デフォルト　クライアントが Embedded SQL、ODBC、OLE DB、または ADO 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。

説明　TIME_ZONE_ADJUSTMENT オプション値は `connection_property('TimeZoneAdjustment')` によって戻される値と同じです。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」923 ページ

TIMESTAMP_FORMAT オプション [互換性]

機能　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)

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。

デフォルト　*YYYY-MM-DD HH:NN:ss.SSS*

Open Client 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。

説明　フォーマットは次の記号を組み合わせた文字列です。

記号	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	Am か Pm (12 時間表記)
pp	必要であれば pm (12 時間表記)
f	フランス語の日と月 (旧式)

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、フォーマットで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、フォーマットで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 Adaptive Server 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)、ゼロ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。

TRUNCATE_DATE_VALUES オプション [データベース] (旧式)

機能　DATE データ型の値にある時刻値の保存形式を変更します。

指定可能な値　**ON**、**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。

デフォルト　バージョン 7 以降のソフトウェアで作成されたデータベース、またはバージョン 7 以降にアップグレードされたデータベースでは **ON**

古いデータベースでは **OFF**

説明

注意
このオプションは推奨されなくなりました。

このオプションが **ON** に設定されると、DATE データ型として宣言されたカラムや変数には、時間と分が保存されません。

オプションを **OFF** に設定すると、DATE カラムや変数には時間と分も保存されます。これらの時間に関するコンポーネントは、DATE_FORMAT の設定があるために表示されません。ただし、DATE 値同士の完全比較は、時間のコンポーネントの不一致により予期せずに失敗することがあります。

参照　「DATE_FORMAT オプション [互換性]」836 ページ

『ASA SQL リファレンス・マニュアル』> 「DATE データ型 [日付と時刻]」

TRUNCATE_TIMESTAMP_VALUES オプション [データベース]

機能 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**

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。

デフォルト **OFF**

[説明] Adaptive Server Anywhere の TIMESTAMP 値の精度は、小数点以下 6 桁までです。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、Adaptive Server 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 同期の場合、最初の同期の実行前にこのオプションを設定してください。

データベース・サーバが TRUNCATE_TIMESTAMP_VALUES および 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。

例　たとえば、DEFAULT_TIMESTAMP_INCREMENT オプションを 100 000 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]

機能　TRUNCATE TABLE 文の処理を高速化します。

指定可能な値　**ON**、**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。

デフォルト　**ON**

説明 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 『ASA SQL リファレンス・マニュアル』> 「TRUNCATE TABLE 文」

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。

TSQL_HEX_CONSTANT オプション [互換性]

機能 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**

説明　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_VARIABLES オプション [互換性]

機能　@ 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**

説明　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。

UPDATE_STATISTICS オプション [データベース]

機能　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**

説明　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・ヒストグラムを自己チューニングします。UPDATE_STATISTICS オプションを **OFF** に設定してクエリ実行中の統計情報の収集を無効にできます。

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。

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。

USER_ESTIMATES オプション [データベース]

機能

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**

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。

デフォルト

OVERRIDE-MAGIC

説明

Adaptive Server 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、『ASA SQL リファレンス・マニュアル』> 「明示的な選択性推定」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。

VERIFY_ALL_COLUMNS オプション [レプリケーション]

機能　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**

説明　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。

VERIFY_THRESHOLD オプション [レプリケーション]

機能　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　**1 000**

説明　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スレッショルドより長い場合は、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。

WAIT_FOR_COMMIT オプション [データベース]

機能　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**

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。

デフォルト **OFF**

説明 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]

関数 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 *hostname-string* または **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。

デフォルト **NULL**

説明 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 および 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 17 章

データベースのパフォーマンスと接続プロパティ

この章の内容

この章では、データベース・パフォーマンスのモニタリングとデータベース接続パラメータに関する情報と表を示します。

データベース・パフォーマンスに関する統計情報

Adaptive Server Anywhere には、データベースのパフォーマンスをモニタするのに使用される各種の統計値を提供します。それらは Sybase Central からアクセスでき、クライアント・アプリケーションからは関数として呼び出せます。さらにサーバによって Windows パフォーマンス・モニタに提供されます。

注意
Windows パフォーマンス・モニタは、Windows NT/2000/XP で使用できます。

ここでは、クライアント・アプリケーションからパフォーマンス関連の統計値にアクセスする方法、Sybase Central を使ってパフォーマンスをモニタする方法、Windows パフォーマンス・モニタを使ってパフォーマンスをモニタする方法について説明します。

パフォーマンス・モニタの統計値

Windows パフォーマンス・モニタはプロセッサ、メモリ、アプリケーションなどのオブジェクトの動作を表示するためのアプリケーションです。Adaptive Server Anywhere はパフォーマンス・モニタが表示する多くの統計値を提供します。

パフォーマンス・モニタの使い方については、『ASA SQL ユーザーズ・ガイド』>「Windows パフォーマンス モニタからのデータベース統計値のモニタリング」を参照してください。

Adaptive Server Anywhere によって、パフォーマンス・モニタは統計値を利用できるようになります。処理速度は毎秒レポートされます。統計値は、次のように分類されています。

- 「キャッシュの統計値」915 ページ
- 「チェックポイントとリカバリの統計値」916 ページ
- 「通信の統計値」917 ページ

キャッシュの統計値　これらの統計値により、キャッシュの使用状態を分析できます。

統計情報	スコープ	説明
キャッシュ・ヒット／秒	接続とデータベース	ルックアップの対象となるデータベース・ページがキャッシュ内で検出された頻度
キャッシュ読み込み：インデックス内部／秒	接続とデータベース	インデックスの内部ノードのページがキャッシュから読み込まれる頻度
キャッシュ読み込み：インデックス・リーフ／秒	接続とデータベース	インデックス・リーフ・ページがキャッシュから読み込まれる頻度
キャッシュ読み込み：テーブル／秒	接続とデータベース	テーブル・ページがキャッシュから読み込まれる頻度
キャッシュ読み込み：合計ページ数／秒	接続とデータベース	データベースのページをキャッシュで検索する頻度
キャッシュ・サイズ：現在値	サーバ	データベース・サーバ・キャッシュの現在のサイズ (キロバイト)
キャッシュ・サイズ：最大値	サーバ	データベース・サーバ・キャッシュの許容最大サイズ (キロバイト)

統計情報	スコープ	説明
キャッシュ・サイズ：最小値	サーバ	データベース・サーバ・キャッシュの許容最小サイズ (キロバイト)
キャッシュ・サイズ：ピーク値	サーバ	データベース・サーバ・キャッシュのピーク時のサイズ (キロバイト)

チェックポイントとリカバリの統計値

これらの統計値により、データベースがアイドル状態のときに行われたチェックポイントとリカバリ動作を分析できます。

統計情報	スコープ	説明
チェックポイント・フラッシュ／秒	データベース	チェックポイントの間に隣接ページを書き出す頻度
チェックポイント・ログ	データベース	トランザクション・ログに対してチェックポイントが行われる頻度
チェックポイントの緊急度	データベース	チェックポイントの緊急度 (パーセント)
チェックポイント／秒	データベース	チェックポイントを実行する頻度
アイドル・アクティブ／秒	データベース	サーバのアイドル・スレッドがアクティブになって、アイドル書き込み、アイドル・チェックポイントなどを行う頻度
アイドル・チェックポイント時間	データベース	アイドル・チェックポイントに費やされた合計時間 (秒)
アイドル・チェックポイント／秒	データベース	チェックポイントがサーバのアイドル・スレッドによって最後まで行われる頻度。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。
アイドル書き込み／秒	データベース	サーバのアイドル・スレッドによってディスク書き込みが発行される頻度

統計情報	スコープ	説明
予想リカバリ I/O 数	データベース	データベースのリカバリに必要な I/O 操作の推定回数
リカバリの緊急度	データベース	リカバリの緊急度 (パーセント)

通信の統計値

これらの統計値により、クライアント／サーバ間の通信状況を分析できます。

統計情報	スコープ	説明
通信：バッファ・ミス	サーバ	接続バッファ・プールを超えている、ネットワーク・バッファ割り付けの合計数
通信：受信バイト数／秒	接続とサーバ	ネットワーク・データが (バイト単位で) 受信される速度
通信：未圧縮状態での受信バイト数／秒	接続とサーバ	圧縮が無効になっていた場合のバイトの受信速度
通信：送信バイト数／秒	接続とサーバ	ネットワークでバイトが送信される速度
通信：未圧縮状態での送信バイト数／秒	接続とサーバ	圧縮が無効になっていた場合のバイトの送信速度
通信：フリー・バッファ	サーバ	空いているネットワーク・バッファの数
通信：使用中ライセンス	サーバ	接続されているユニークなクライアント・ネットワーク・アドレス数
通信：受信マルチパケット数／秒	サーバ	マルチパケット・デリバリの受信速度
通信：送信マルチパケット数／秒	サーバ	マルチパケット・デリバリの送信速度
通信：受信パケット数／秒	接続とサーバ	ネットワーク・パケットの受信速度

統計情報	スコープ	説明
通信：未圧縮状態での受信パケット数／秒	接続とサーバ	圧縮が無効になっていた場合のネットワーク・パケットの受信速度
通信：送信パケット数／秒	接続とサーバ	ネットワーク・パケットの送信速度
通信：未圧縮状態での送信パケット数／秒	接続とサーバ	圧縮が無効になっていた場合のネットワーク・パケットの送信速度
通信：失敗した送信／秒	サーバ	基本のプロトコルがパケット送信に失敗した頻度
通信：合計バッファ	サーバ	ネットワーク・バッファの総数

ディスク I/O の統計値

これらの統計値により、ディスクへのアクセス (読み込みと書き込み) 状況を取得し、ディスク I/O に使用されたアクティビティの負荷について全体的な情報を把握できます。

統計情報	スコープ	説明
ディスク：アクティブ I/O	データベース	サーバが発行する、まだ完了していないファイル I/O の現在の数
ディスク：最大 I/O	データベース	[ディスク読み込み：アクティブ I/O] が到達した最大値

ディスク読み込みの統計値

これらの統計値により、ディスクから情報を読み込むのに使用されたアクティビティの負荷とそのタイプを分析できます。

統計情報	スコープ	説明
ディスク読み込み：合計ページ数／秒	接続とデータベース	ページがファイルから読み込まれる速度
ディスク読み込み：アクティブ	データベース	サーバによって発行される、まだ完了していないファイル読み込みの現在の数

統計情報	スコープ	説明
ディスク読み込み：インデックス内部／秒	接続とデータベース	インデックス内部ノードのページがディスクから読み込まれる頻度
ディスク読み込み：インデックス・リーフ／秒	接続とデータベース	インデックス・リーフ・ページをディスクから読み込む頻度
ディスク読み込み：テーブル／秒	接続とデータベース	テーブル・ページがディスクから読み込まれる頻度
ディスク読み込み：アクティブの最大値	データベース	[ディスク読み込み：アクティブ] が到達した最大値

ディスク書き込みの統計値

これらの統計値により、ディスクへ情報を書き込むのに使用されたアクティビティの負荷とそのタイプを分析できます。

統計情報	スコープ	説明
ディスク書き込み：アクティブ	データベース	サーバが発行する、まだ完了していないファイル書き込みの現在の数
ディスク書き込み：アクティブの最大値	データベース	[ディスク書き込み：アクティブ] が到達した最大値
ディスク書き込み：コミット・ファイル／秒	データベース	サーバが強制的に行うディスク・キャッシュのフラッシュの頻度。Windows NT/2000/XP と NetWare プラットフォームではバッファなし (direct) の IO が使われるため、フラッシュは不要です。
ディスク書き込み：データベース拡張／秒	データベース	データベース・ファイルの拡張頻度 (ページ／秒)
ディスク書き込み：テンポラリ拡張／秒	データベース	テンポラリ・ファイルの拡張頻度 (ページ／秒)
ディスク書き込み：ページ／秒	接続とデータベース	修正されたページがディスクに書き込まれる頻度

統計情報	スコープ	説明
ディスク書き込み：トランザクション・ログ／秒	接続とデータベース	ページをトランザクション・ログに書き込む頻度
ディスク書き込み：トランザクション・ログ・グループのコミット	接続とデータベース	トランザクション・ログのコミットが要求されたときすでにログが書き込まれている (コミットはいつでも可能) 場合に発生

インデックスの統計値

これらの統計値により、インデックスの使用状態を分析できます。

統計情報	スコープ	説明
インデックス：追加／秒	接続とデータベース	インデックスにエントリが追加される頻度
インデックス：ルックアップ／秒	接続とデータベース	インデックスでエントリを検索する頻度
インデックス：完全比較／秒	接続とデータベース	インデックスのハッシュ値を超える比較が必要となる頻度

Java VM の統計値

これらの統計値により、Java VM が使用するメモリ状況を分析できます。

統計情報	スコープ	説明
JVM：グローバル固定サイズ	サーバ	Java VM 固定ヒープに割り付けられる合計バイト数
JVM：ヒープ・サイズ	接続とデータベース	接続ごとに Java VM に割り付けられた合計バイト数
JVM：ネームスペース・サイズ	データベース	Java VM のネームスペースに割り付けられた合計バイト数

メモリ・ページの統計値

これらの統計値により、データベース・サーバが使用しているメモリ量とその目的を分析できます。

統計情報	スコープ	説明
メモリ・ページ：ロック・ヒープ	サーバ	キャッシュでロックされているヒープ・ページの数
メモリ・ページ：ロック・テーブル	データベース	ロック情報の保持に使用されているページ数
メモリ・ページ：ロールバック・ログ	接続とデータベース	ロールバック・ログのページ数
メモリ・ページ：メイン・ヒープ	サーバ	グローバル・サーバ・データ構造に使用されたページ数
メモリ・ページ：マップ・ページ	データベース	ロック・テーブル、頻出するテーブル、テーブル・レイアウトへのアクセスに使用されたマップ・ページ数
メモリ・ページ：プロシージャ定義	データベース	プロシージャに使用された再配置可能なヒープ・ページ数
メモリ・ページ：再配置可能	データベース	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
メモリ・ページ：再配置／秒	データベース	再配置可能なヒープ・ページがテンポラリ・ファイルから読み出される頻度
メモリ・ページ：トリガ定義	データベース	トリガで使用される再配置可能なヒープ・ページの数
メモリ・ページ：ビュー定義	データベース	ビューで使用される再配置可能なヒープ・ページの数

要求の統計値

これらの統計値により、クライアント・アプリケーションの要求への応答に使用されたデータベース・サーバのアクティビティを分析できます。

統計情報	スコープ	説明
要求	サーバ	サーバが新しい要求を処理するか、既存の要求の処理を続行できる状態になる頻度

統計情報	スコープ	説明
要求：アクティブ	サーバ	現在、要求を処理しているサーバ・スレッドの数
要求：未スケジュール	サーバ	使用できるサーバ・スレッドが空くのを待ってキューイングされている要求の数
カーソル	接続	現在サーバが保持している宣言されたカーソルの数
開いているカーソル	接続	現在サーバが保持しているオープン・カーソルの数
文	接続	現在サーバが保持している準備文の数
文の準備	接続	サーバにより文の準備が処理される頻度
トランザクションのコミット	接続	コミット要求が処理される頻度
トランザクションのロールバック	接続	ロールバック要求が処理される頻度

その他の統計値

統計情報	スコープ	説明
使用可能 IO	サーバ	カウントとして表示
接続カウント	データベース	このデータベースとの接続の数
メイン・ヒープ・バイト	サーバ	グローバル・サーバ・データ構造に使用されたバイト数
メモリ不足クエリ方法	接続とデータベース	メモリ不足状態のため、サーバがその実行中に実行プランを変更した回数
テンポラリ・テーブル・ページ	接続とデータベース	テンポラリ・テーブルで使用されるテンポラリ・ファイル内のページ数

データベース・プロパティ

Adaptive Server Anywhere 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を提供します。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、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 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。

説明

プロパティ	説明
Allow_nulls_by_default	「ALLOW_NULLS_BY_DEFAULT オプション [互換性]」819 ページ
Ansi_blanks	「ANSI_BLANKS オプション [互換性]」819 ページ
Ansi_close_cursors_on_rollback	「ANSI_CLOSE_CURSORS_ON_ROLLBACK オプション [互換性]」820 ページ
Ansi_integer_overflow	「ANSI_INTEGER_OVERFLOW オプション [互換性]」820 ページ
Ansi_permissions	「ANSI_PERMISSIONS オプション [互換性]」821 ページ
Ansi_update_constraints	「ANSI_UPDATE_CONSTRAINTS オプション [互換性]」822 ページ
Ansinull	「ANSINULL オプション [互換性]」823 ページ

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、または ADO.NET 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]」237 ページ を参照してください。
Auditing	「AUDITING オプション [データベース]」824 ページ
AuditingTypes	現在有効な監査タイプ。「AUDITING オプション [データベース]」824 ページ
Automatic_timestamp	「AUTOMATIC_TIMESTAMP オプション [互換性]」826 ページ
Background_priority	「BACKGROUND_PRIORITY オプション [データベース]」826 ページ
BlockedOn	現在の接続が制限されていない場合は 0。ブロックされている場合は、ロック矛盾によってブロックされる接続の数。
Blocking	「BLOCKING オプション [データベース]」828 ページ
Blocking_timeout	「BLOCKING_TIMEOUT オプション [データベース]」828 ページ
BytesReceived	クライアント／サーバ通信中に受信したバイト数
BytesReceivedUncomp	圧縮が無効になっていた場合にクライアント／サーバ通信中に受信されるバイト数 (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したバイト数

プロパティ	説明
BytesSentUncomp	圧縮が無効になっていた場合にクライアント／サーバ通信中に送信されるバイト数 (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数
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
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
CacheReadTable	キャッシュから読み込まれたテーブル・ページの数
Chained	「CHAINED オプション [互換性]」829 ページ
CharSet	接続で使用される文字セット
Checkpoint_time	「CHECKPOINT_TIME オプション [データベース]」829 ページ
Cis_option	「CIS_OPTION オプション [データベース]」830 ページ
Cis_rowset_size	予約
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、および iAnywhere JDBC ドライバ接続の場合は **CmdSeq** を返す。
ClientPort	クライアントの TCP/IP ポート番号を返します。また、TCP/IP 接続でない場合は **0** を返します。
Close_on_EndTrans	「CLOSE_ON_ENDTRANS オプション [互換性]」831 ページ
Commit	処理されたコミット要求の数

プロパティ	説明
CommLink	接続用の通信リンク。Adaptive Server Anywhere がサポートしているネットワーク・プロトコル、または同一マシン接続の場合は **local**。
CommNetworkLink	接続用の通信リンク。これは、Adaptive Server Anywhere がサポートするネットワーク・プロトコルの 1 つです。値は、**SharedMemory**、**TCPIP**、**SPX**、または **NamedPipes** です。CommLinkNetwork プロパティは、同一マシン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**、OLE DB 接続、ADO.NET 接続、および iAnywhere JDBC ドライバ接続の場合は **CmdSeq** を返す。
Compression	この接続で通信圧縮が有効であるかどうかを示す ON または OFF を返す
Connection_authentication	クライアントを認証するために使用する文字列。データベースが変更可能になる前に、認証が必要です。
Conversion_error	「CONVERSION_ERROR オプション [互換性]」834 ページ
Cooperative_commit_timeout	「COOPERATIVE_COMMIT_TIMEOUT オプション [データベース]」835 ページ
Cooperative_commits	「COOPERATIVE_COMMITS オプション [データベース]」835 ページ
Cursor	現在サーバによって保守されている宣言されたカーソルの数
CursorOpen	現在サーバによって管理されているオープン・カーソルの数
Database_authentication	データベースを認証するために使用する文字列。データベースが変更可能になる前に、認証が必要です。
Date_format	「DATE_FORMAT オプション [互換性]」836 ページ
Date_order	「DATE_ORDER オプション [互換性]」838 ページ

プロパティ	説明
DBNumber	データベースの ID 番号
Debug_messages	「DEBUG_MESSAGES オプション [データベース]」 839 ページ
Dedicated_task	「DEDICATED_TASK オプション [データベース]」 840 ページ
Default_timestamp_increment	「DEFAULT_TIMESTAMP_INCREMENT オプション [データベース]」 841 ページ
Delayed_commit_timeout	「DELAYED_COMMIT_TIMEOUT オプション [データベース]」 842 ページ
Delayed_commits	「DELAYED_COMMITS オプション [データベース]」 843 ページ
DiskRead	ディスクから読み込まれたページ数
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
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
DiskReadTable	ディスクから読み込まれたテーブル・ページの数
DiskWrite	ディスクへ書き込まれた修正ページの数
Divide_by_zero_error	「DIVIDE_BY_ZERO_ERROR オプション [互換性]」 845 ページ
Encryption	「Encryption 接続パラメータ [ENC]」 256 ページ
Escape_character	「ESCAPE_CHARACTER オプション [互換性]」 846 ページ
EventName	接続でイベント・ハンドラが実行されている場合の関連するイベント名。それ以外の場合、結果は NULL。
Exclude_operators	「EXCLUDE_OPERATORS オプション [データベース]」 846 ページ

プロパティ	説明
Extended_join_syntax	「EXTENDED_JOIN_SYNTAX オプション [データベース]」 846 ページ
Fire_triggers	「FIRE_TRIGGERS オプション [互換性]」 847 ページ
First_day_of_week	「FIRST_DAY_OF_WEEK オプション [データベース]」 847 ページ
Float_as_double	「FLOAT_AS_DOUBLE オプション [互換性]」 848 ページ
For_xml_null_treatment	「FOR_XML_NULL_TREATMENT オプション [データベース]」 849 ページ
Force_view_creation	「FORCE_VIEW_CREATION オプション [データベース]」 850 ページ
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
Global_database_id	「GLOBAL_DATABASE_ID オプション [データベース]」 850 ページ
IdleTimeout	接続のアイドル・タイムアウト値 詳細については、「Idle 接続パラメータ [IDLE]」 263 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
IndLookup	インデックスの中で検索されたエントリの数
Integrated_server_name	「INTEGRATED_SERVER_NAME オプション [データベース]」 853 ページ
Isolation_level	「ISOLATION_LEVEL オプション [互換性]」 853 ページ
Java_heap_size	「JAVA_HEAP_SIZE オプション [データベース]」 860 ページ
Java_input_output	「JAVA_INPUT_OUTPUT オプション [データベース]」 860 ページ

プロパティ	説明
Java_namespace_size	「JAVA_NAMESPACE_SIZE オプション [データベース]」 861 ページ
Java_page_buffer_size	Java VM によって使用されるページ・バッファ・サイズ
JavaHeapSize	Java VM あたりのヒープ・サイズ
Language	ロケール言語
LastIdle	要求間のチックの数
LastReqTime	指定の接続に対する最後の要求が開始した時間
LastStatement	現在の接続で最後に作成された SQL 文。 詳細については、「-zl サーバ・オプション」 223 ページ を参照してください。
LivenessTimeout	現在の接続の活性タイムアウト時間。 詳細については、「LivenessTimeout 接続パラメータ [LTO]」 267 ページ を参照してください。
Lock_rejected_rows	予約
LockName	64 ビット符号なし整数。接続が待機しているロックを示します。
Log_deadlocks	「LOG_DEADLOCKS オプション [データベース]」 861 ページ
LogFreeCommit	Redo Free Commit の数。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。
Login_mode	「LOGIN_MODE オプション [データベース]」 862 ページ
Login_procedure	「LOGIN_PROCEDURE オプション [データベース]」 863 ページ
LoginTime	接続が確立された日付と時刻。

プロパティ	説明
LogWrite	トランザクション・ログに書き込まれたページの数
Max_cursor_count	「MAX_CURSOR_COUNT オプション [データベース]」866 ページ
Max_plans_cached	「MAX_PLANS_CACHED オプション [データベース]」867 ページ
Max_recursive_iterations	「MAX_RECURSIVE_ITERATIONS オプション [データベース]」868 ページ
Max_statement_count	「MAX_STATEMENT_COUNT オプション [データベース]」868 ページ
MessageReceived	MESSAGE 文によって生成され、WAITFOR 文を中断させた文字列。それ以外の場合は、空の文字列が返されます。
Min_password_length	「MIN_PASSWORD_LENGTH オプション [データベース]」870 ページ
Name	現在の接続の名前
Nearest_century	「NEAREST_CENTURY オプション [互換性]」870 ページ
NodeAddress	クライアント／サーバ接続のクライアント用ノード クライアントとサーバの両方が同じマシンにある場合は空の文字列が返されます。
Non_keywords	「NON_KEYWORDS オプション [互換性]」871 ページ
Number	接続の ID 番号
ODBC_describe_binary_as_varbinary	「ODBC_DESCRIBE_BINARY_AS_VARBINARY [データベース]」872 ページ
ODBC_distinguish_char_and_varchar	「ODBC_DISTINGUISH_CHAR_AND_VARCHAR オプション [データベース]」873 ページ
On_charset_conversion_failure	「ON_CHARSET_CONVERSION_FAILURE オプション [データベース]」874 ページ

プロパティ	説明
On_tsql_error	「ON_TSQL_ERROR オプション [互換性]」875 ページ
Optimistic_wait_for_commit	「OPTIMISTIC_WAIT_FOR_COMMIT オプション [互換性]」876 ページ
Optimization_goal	「OPTIMIZATION_GOAL オプション [データベース]」877 ページ
Optimization_level	予約
Optimization_workload	「OPTIMIZATION_WORKLOAD オプション [データベース]」879 ページ
PacketSize	接続で使用されるバイト単位のパケット・サイズ
PacketsReceived	受信したクライアント／サーバ通信パケットの数
PacketsReceivedUncomp	圧縮が無効になっていた場合にクライアント／サーバ通信中に受信されるパケット数 (この値は、圧縮が無効の場合は PacketsReceived の値と同じ)
PacketsSent	送信したクライアント／サーバ通信パケットの数
PacketsSentUncomp	圧縮が無効になっていた場合にクライアント／サーバ通信中に送信されるパケット数 (この値は、圧縮が無効の場合は PacketsSent の値と同じ)
Percent_as_comment	「PERCENT_AS_COMMENT オプション [互換性]」883 ページ
Pinned_cursor_percent_of_cache	「PINNED_CURSOR_PERCENT_OF_CACHE オプション [データベース]」884 ページ
Precision	「PRECISION オプション [データベース]」884 ページ
Prefetch	「PREFETCH オプション [データベース]」885 ページ
Prepares	実行された文の準備作業の数
PrepStmt	現在サーバによって管理されている準備文の数

プロパティ	説明
Preserve_source_format	「PRESERVE_SOURCE_FORMAT オプション [データベース]」886 ページ
Prevent_article_pkey_update	「PREVENT_ARTICLE_PKEY_UPDATE オプション [データベース]」887 ページ
Query_plan_on_open	「QUERY_PLAN_ON_OPEN オプション [互換性]」888 ページ
QueryBypassed	オプティマイザを利用せずに最適化された要求の数
QueryCachedPlans	接続毎で現在キャッシュされている実行プラン数
QueryCachePages	キャッシュ内で実行プランが保存されるページ数
QueryLowMemoryStrategy	メモリ不足状態のため、サーバがその実行中に実行プランを変更した回数。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	最適化された要求の数
QueryReused	プラン・キャッシュから再利用された要求の数
Quoted_identifier	「QUOTED_IDENTIFIER オプション [互換性]」889 ページ
Read_past_deleted	「READ_PAST_DELETED オプション [データベース]」890 ページ
Recovery_time	「RECOVERY_TIME オプション [データベース]」890 ページ
Remote_idle_timeout	「REMOTE_IDLE_TIMEOUT オプション [データベース]」891 ページ
Replicate_all	「REPLICATE_ALL オプション [レプリケーション]」891 ページ
ReqType	最後の要求のタイプを示す文字列

プロパティ	説明
Return_date_time_as_string	「RETURN_DATE_TIME_AS_STRING オプション [データベース]」892 ページ
RI_trigger_time	「RI_TRIGGER_TIME オプション [互換性]」894 ページ
Rlbk	処理されたロールバック要求の数
Rollback_on_deadlock	「ROLLBACK_ON_DEADLOCK [データベース]」894 ページ
RollbackLogPages	ロールバック・ログのページ数
Row_counts	「ROW_COUNTS オプション [データベース]」895 ページ
Scale	「SCALE オプション [データベース]」895 ページ
ServerPort	サーバの TCP/IP ポート番号または **0** を返す
Sort_collation	「SORT_COLLATION オプション [データベース]」896 ページ
SQL_flagger_error_level	「SQL_FLAGGER_ERROR_LEVEL オプション [互換性]」897 ページ
SQL_flagger_warning_level	「SQL_FLAGGER_WARNING_LEVEL オプション [互換性]」897 ページ
String_rtruncation	「STRING_RTRUNCATION オプション [互換性]」898 ページ
Subsume_row_locks	「SUBSUME_ROW_LOCKS オプション [データベース]」899 ページ
Suppress_TDS_debugging	「SUPPRESS_TDS_DEBUGGING オプション [データベース]」900 ページ
TDS_empty_string_is_null	「TDS_EMPTY_STRING_IS_NULL オプション [データベース]」900 ページ

プロパティ	説明
Temp_space_limit_check	「TEMP_SPACE_LIMIT_CHECK オプション [データベース]」 901 ページ
TempTablePages	テンポラリ・テーブルで使用されるテンポラリ・ファイル内のページ数
Time_format	「TIME_FORMAT オプション [互換性]」 902 ページ
Timestamp_format	「TIMESTAMP_FORMAT オプション [互換性]」 903 ページ
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。デフォルトでは、クライアントのタイム・ゾーンに応じて設定されます。
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間を含む文字列、または最後の 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
Truncate_date_values	「TRUNCATE_DATE_VALUES オプション [データベース] (旧式)」 905 ページ
Truncate_timestamp_values	「TRUNCATE_TIMESTAMP_VALUES オプション [データベース]」 906 ページ
Truncate_with_autocommit	「TRUNCATE_WITH_AUTO_COMMIT オプション [データベース]」 907 ページ
Tsql_hex_constant	「TSQL_HEX_CONSTANT オプション [互換性]」 908 ページ
Tsql_variables	「TSQL_VARIABLES オプション [互換性]」 909 ページ
UncommitOp	コミットされていないオペレーションの数
User_estimates	「USER_ESTIMATES オプション [データベース]」 910 ページ

プロパティ	説明
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。 詳細については、「AppInfo 接続パラメータ [APP]」237 ページ を参照してください。
Userid	接続のユーザ ID
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について ON または OFF を返す。 詳細については、「-gu サーバ・オプション」197 ページを参照してください。
Wait_for_commit	「WAIT_FOR_COMMIT オプション [データベース]」911 ページ

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、次の文はメイン・ヒープの保持に使用されるキャッシュ・ページ数を戻します。

```
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
```

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
AvailIO	予約
BuildChange	予約
BuildClient	予約
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
BytesReceivedUncomp	圧縮が無効になっていた場合にクライアント／サーバ通信中に受信されるバイト数 (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したバイト数
BytesSentUncomp	圧縮が無効になっていた場合にクライアント／サーバ通信中に送信されるバイト数 (この値は、圧縮が無効の場合は BytesSent の値と同じ)
C2	サーバの起動時に -sc オプションが使用された場合は YES を返す。それ以外の場合は、NO を返します。 詳細については、「-sc サーバ・オプション」210 ページを参照してください。
CacheHitsEng	データベース・ページのルックアップ回数
CacheReplacements	キャッシュの中で置換されたページの数
CharSet	データベース・サーバが使用する文字セット

プロパティ	説明
CommandLine	サーバを起動するために使用したコマンド・ライン -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。 暗号化キーを指定する必要がある場合、-ep オプションでキーを入力するためにプロンプト表示してデータベースを起動するか、または START DATABASE 文を使用できます。また、データベースが自動的に起動できる場合、キーは DBKEY 接続パラメータで提供されます。
CompactPlatformVer	PlatformVer プロパティの要約版
CompanyName	ソフトウェアを所有している会社の名前
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 ON または OFF を返す。 詳細については、『ASA SQL リファレンス・マニュアル』> 「sa_server_option システム・プロシージャ」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、データベース・サーバ・ウィンドウからのメッセージがログされるファイル名を返す。それ以外は、空の文字列を返します。
CurrentCacheSize	現在のキャッシュ・サイズ (キロバイト)
DefaultCollation	新規データベースに使用される照合 (明示的に指定されている照合がない場合)
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 を返す
FreeBuffers	使用できるネットワーク・バッファの数

プロパティ	説明
IdleTimeout	デフォルトのアイドル・タイムアウト。 詳細については、「-ti サーバ・オプション」210 ページを参照してください。
IsFipsAvailable	FIPS DLL がインストールされていれば YES を返し、それ以外は NO を返す。
IsIQ	サーバが IQ サーバであれば YES を返し、それ以外は NO を返す
IsJavaAvailable	JavaVM がインストールされている場合は YES、JavaVM がインストールされていない場合は NO を返す。このプロパティは、Java VM が使用可能かどうかだけを示し、現在使用中であるかどうかは示されません。
IsNetworkServer	ネットワーク・データベース・サーバに接続した場合は YES、パーソナル・データベース・サーバに接続した場合は NO を返す
IsRuntimeServer	制限のあるデスクトップ・ランタイム・データベース・サーバに接続した場合は YES、それ以外の場合は NO を返す
JavaGlobFix	Java VM のグローバルな固定サイズ
Language	サーバのロケール言語
LegalCopyright	ソフトウェアの著作権を表した文字列
LegalTrademarks	ソフトウェアの商標に関する情報
LicenseCount	ライセンス供与されているシートまたはプロセッサの数
LicensedCompany	ライセンス供与されている会社名
LicensedUser	ライセンス供与されているユーザ名
LicensesinUse	現在同時にネットワーク・サーバに接続しているユーザ数。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・アドレスによって判断します。

プロパティ	説明
LicenseType	ライセンス・タイプ。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
LockedHeapPages	キャッシュでロックされているヒープ・ページの数
MachineName	データベース・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
MainHeapBytes	グローバル・サーバ・データ構造に使用されたバイト数
MainHeapPages	グローバル・サーバ・データ構造に使用されたページ数
MaxCacheSize	許容最大キャッシュ・サイズ (キロバイト)
MaxMessage	サーバのメッセージ・ウィンドウから取得できる行番号の現在の最大値。サーバのメッセージ・ウィンドウに最後に表示されたメッセージを示します。
Message, *linenumber*	サーバのメッセージ・ウィンドウから取り出された行。メッセージの前には、メッセージが表示された日付と時間が付けられます。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。
MessageText, *linenumber*	サーバのメッセージ・ウィンドウで指定した行番号に関連付けられたテキスト。日付と時刻のプレフィクスはありません。2 番目のパラメータは行番号を指定します。

プロパティ	説明
MessageTime, *linenumber*	サーバのメッセージ・ウィンドウで指定した行番号に関連付けられた日付と時刻。2 番目のパラメータは行番号を指定します。
MessageWindowSize	サーバのメッセージ・ウィンドウから取得できる行番号の最大値
MinCacheSize	許容最小キャッシュ・サイズ (キロバイト)
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・デリバリの数
MultiPacketsSent	クライアント／サーバ通信中に送信したマルチパケット・デリバリの数
Name	サーバ名

プロパティ	説明
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) で、ネイティブのプロセッサ・タイプを識別する文字列を返す。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 (CE 以外) ― X86 NetWare ― X86 Intel Solaris ― X86 CE ― SH3、SH4、MIPS、または ARM 64 ビット版 Windows ― IA64 または AMD64 64 ビット UNIX ― IA64 または AMD64 Solaris ― SPARC AIX ― PPC MAC OS ― PPC HP ― PA_RISC DEC UNIX ― ALPHA Linux ― X86、SPARC、IA64
NumProcessorsAvail	サーバにあるプロセッサの数
NumProcessorsMax	使用されるプロセッサの最大数。通常、*dbeng.exe* では 2、*dbsrv.exe* では 0 です。
PacketsReceived	受信したクライアント／サーバ通信パケットの数

プロパティ	説明
PacketsReceivedUncomp	圧縮が無効になっていた場合にクライアント／サーバ通信中に受信されるパケット数 (この値は、圧縮が無効の場合は PacketsReceived の値と同じ)
PacketsSent	送信したクライアント／サーバ通信パケットの数
PacketsSentUncomp	圧縮が無効になっていた場合にクライアント／サーバ通信中に送信されるパケット数 (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。-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到達したキャッシュの最大値 (キロバイト)
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。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。
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。ビルド番号やサービス・パックなどを含みます。たとえば、`Windows 2000 Build 2195 Service Pack 3` などを返します。
ProcessCPU	サーバのプロセスに対する CPU 使用率の統計値。値は秒数で指定します。このプロパティは WindowsNT/2000/XP、Windows95/98/Me、UNIX でサポートされます。Windows CE または NetWare ではサポートされていません。
ProcessCPUSystem	システムによるプロセス CPU の使用状況。値は秒数で指定します。このプロパティは WindowsNT/2000/XP、Windows95/98/Me、UNIX でサポートされます。Windows CE または NetWare ではサポートされていません。

プロパティ	説明
ProcessCPUUser	ユーザによるプロセス CPU の使用状況。値は秒数で指定します。このプロパティは WindowsNT/2000/XP、Windows95/98/Me、UNIX でサポートされます。Windows CE または NetWare ではサポートされていません。
ProcessorArchitecture	プロセッサ・タイプを識別する文字列。値には、次のものがあります。 32 ビット版 Windows (CE 以外) ― X86 NetWare ― X86 Intel Solaris ― X86 CE ― SH3、SH4、または ARM 64 ビット版 Windows ― IA64 または AMD64 64 ビット UNIX ― IA64 または AMD64 Solaris ― SPARC AIX ― PPC MAC OS ― PPC HP ― PA_RISC DEC UNIX ― ALPHA
ProductName	ソフトウェア名

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。 詳細については、『ASA SQL リファレンス・マニュアル』>「sa_server_option システム・プロシージャ」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。
ProductVersion	実行中のソフトウェアのバージョン
QuittingTime	サーバのシャットダウン時間。none を指定すると、値は **none** です。
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 ON を返し、それ以外は OFF を返す。
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る頻度
RequestLogFile	要求ロギング・ファイルの名前。レベルのロギングがない場合は、空の文字列を返します。 詳細については、『ASA SQL リファレンス・マニュアル』>「sa_server_option システム・プロシージャ」を参照してください。
RequestLogging	ALL、SQL、または NONE。 詳細については、『ASA SQL リファレンス・マニュアル』>「sa_server_option システム・プロシージャ」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。 詳細については、『ASA SQL リファレンス・マニュアル』> 「sa_server_option システム・プロシージャ」を参照してください。
SendFail	通信プロトコルがパケット送信に失敗した回数
StartTime	サーバが開始した日付／時刻
Tempdir	サーバに格納されているテンポラリ・ファイルのディレクトリ
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分数
TotalBuffers	ネットワーク・バッファの総数
UnschReq	使用できるサーバ・スレッドが空くのを待ってキューイングされている要求の数

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
```

説明

プロパティ	説明
Alias	データベース名
BlankPadding	ブランク埋め込み機能のステータス。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。
BlobArenas	BlobArenas 機能のステータス。データベースが、拡張 (BLOB) ページをデータベース用のテーブル・ページとは別に保存する場合は ON を返します。それ以外の場合は、OFF を返します。
CacheHits	キャッシュ内検索の際、キャッシュ内のページに存在したデータベース・ページの数
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
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
CacheReadTable	キャッシュから読み込まれたテーブル・ページの数
Capabilities	データベースに対して有効にされる機能ビット。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。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。
CaseSensitivePasswords	パスワードでの大文字と小文字の区別のステータス。バージョン 9.0.0 以降では、パスワードの大文字と小文字の区別は、データベースでの大文字と小文字の区別とは別です。データベース・パスワード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。
CharSet	データベースの文字セット

プロパティ	説明
CheckpointUrgency	データベースのチェックポイント時間設定の割合として表した、直前のチェックポイントからの経過時間
Checksum	データベースのページ・チェックサムがデータベースで有効な場合は、ON を返す。それ以外の場合は、OFF を返します。
Chkpt	実行されたチェックポイントの数
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
ChkptPage	トランザクション・ログ・チェックポイントの数
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。Windows NT/2000/XP と NetWare プラットフォームでは、バッファなし (ダイレクトな) IO が使用される場合は、ディスク・キャッシュのフラッシュは不要です。
CompressedBTrees	圧縮された B ツリー・インデックスがサポートされている場合は ON を返す。それ以外の場合は、OFF を返します。
Compression	データベースの圧縮ステータス。ON (データベースは圧縮されている) または OFF を返す。圧縮データベース上にライト・ファイルを作成しても圧縮されません。圧縮データベースに作成したライト・ファイルを開いて db_property('icompression') を選択すると、OFF が返されます。
ConnCount	データベースとの接続の数
CurrentRedoPos	次のデータベース操作が記録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
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 数
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の数

プロパティ	説明
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の数
DBFileFragments	データベース・ファイルのフラグメント数。Windows NT/2000/XP のみ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数
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
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
DiskReadTable	ディスクから読み込まれたテーブル・ページの数
DiskWrite	ディスクへ書き込まれた修正ページの数
DriveType *dbspace*	データベース・ファイルが配置されているドライブ。CD、FIXED、RAMDISK、REMOTE、REMOVABLE、UNKNOWN を返します。 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。 **db_extended_property** とともに使用する場合、サイズを知りたい DB 領域を指定できます。 *Dbspace* には、DB 領域の *name* または DB 領域の *file_id* を指定できます。 *dbspace* を指定しないままにするか、*system* を使用すると、システム DB 領域を参照します。 指定した DB 領域が存在しない場合、プロパティ関数が NULL を返します。DB 領域名が指定されて、現在接続されてないデータベースの ID が指定されている場合も、この関数が NULL を返します。

プロパティ	説明
Encryption	データベースに適用されている暗号化のタイプ。None、Simple、または AES を返します。
ExtendDB	データベース・ファイルを拡張したページの数
ExtendTempWrite	テンポラリ・ファイルを拡張したページの数
ファイル	データベース・ルート・ファイルのパスを含む名前
FileSize *dbspace*	**db_property** とともに使用する場合、システム DB 領域のファイル・サイズをページ数で返す。 **db_extended_property** とともに使用する場合、サイズを知りたい DB 領域を指定できます。 *Dbspace* には、DB 領域の *name*、*file_id*、またはテンポラリ DB 領域を参照するための *temporary* を指定できます。 また、*translog* を指定して、ログ・ファイルのサイズを返すこともできます。 さらに、*writefile* を指定して、ライト・ファイルを参照することもできます。ライト・ファイルを使用する場合、DB 領域の FileSize が仮想 DB 領域の総スペースを返します。これは、基本の DB 領域にライト・ファイルに保存された DB 領域に対する変更分を加えたものになります。 *dbspace* を指定しないままにするか、*system* を使用すると、システム DB 領域を参照します。 指定した DB 領域が存在しない場合、プロパティ関数が NULL を返します。DB 領域名が指定されて、現在接続されていないデータベースの ID または名前が指定されている場合も、この関数が NULL を返します。
FileVersion	データベース・ファイルのバージョン。ソフトウェアのリリース・バージョンには対応していません。

プロパティ	説明
FreePageBitMaps	空いているデータベース・ページをビットマップを使用して管理する場合は ON を返す。それ以外の場合は、OFF を返します。
FreePages *dbspace*	FreePages がサポートされているのは、バージョン 8.0.0 以降で作成したデータベースのみ。 **db_property** とともに使用する場合、システム DB 領域の空きページの数を返します。 **db_extended_property** とともに使用する場合、DB 領域の空きページ数を指定できます。*Dbspace* には、DB 領域の *name*、*file_id*、またはテンポラリ DB 領域を参照するための *temporary* を指定できます。 また、*translog* を指定して、ログ・ファイルの空きページ数を返すこともできます。 さらに、*writefile* を指定して、ライト・ファイルを参照することもできます。ライト・ファイルを使用する場合、DB 領域の FreePages が仮想 DB 領域の空きページ数を返します。これは、基本の DB 領域にライト・ファイルに保存された DB 領域に対する変更分を加えたものになります。 *dbspace* を指定しないままにするか、*system* を使用すると、システム DB 領域を参照します。 指定した DB 領域が存在しない場合、プロパティ関数が null を返します。DB 領域名が指定されて、現在接続されていないデータベースの ID または名前が指定されている場合も、この関数が null を返します。
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
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

プロパティ	説明
Histograms	オプティマイザ統計がヒストグラムとして保持されている場合は ON を返す。それ以外の場合は、OFF を返します。
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
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。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
IdleWrite	サーバのアイドル・スレッドが出したディスク書き込みの数
IndAdd	インデックスに追加されたエントリの数
IndLookup	インデックスの中で検索されたエントリの数
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
IQStore	予約
JavaHeapSize	Java VM あたりのヒープ・サイズ
JavaNSSize	Java VM ネームスペースのサイズ
JDKVersion	このデータベースで使用される Java ランタイム・ライブラリのバージョン
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す。言語は 2 文字の ISO フォーマットです。言語が不明のもの (通常はカスタム照合) である場合は、戻り値は NULL です。 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」434 ページを参照してください。

プロパティ	説明
LargeProcedureIDs	このデータベースで 32 ビットのストアド・プロシージャ ID がサポートされている場合は、ON を返す。それ以外の場合は、OFF 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数
LogFileFragments	ログ・ファイルのフラグメント数。Windows NT/2000/XP のみでサポートされています。
LogFreeCommit	Redo Free Commit の数。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。
LogName	トランザクション・ログのパスを含むファイル名
LogWrite	トランザクション・ログに書き込まれたページの数
LTMGeneration	LTM または Replication Agent の世代番号。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
MapPages	ロック・テーブル、頻出するテーブル、テーブル・レイアウトへのアクセスに使用されたマップ・ページ数
MaxIO	CurrIO が到達した最大値
MaxRead	CurrRead が到達した最大値
MaxWrite	CurrWrite が到達した最大値
MultiByteCharSet	データベースでマルチバイト文字セットを使用する場合は、ON を返す。それ以外の場合は、OFF を返します。
Name	データベース名 (エイリアスと同じ)
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

プロパティ	説明
PageSize	データベースのページ・サイズ (バイト)
PreserveSource	データベースがプロシージャとビューのソースを保護する場合は、ON を返す。それ以外の場合は、OFF を返します。
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
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な場合は、ON を返す。それ以外の場合は、OFF を返します。
QueryBypassed	オプティマイザを利用せずに最適化された要求の数
QueryCachedPlans	接続毎で現在キャッシュされている実行プラン数
QueryCachePages	キャッシュ内で実行プランが保存されるページ数
QueryLowMemoryStrategy	メモリ不足状態のため、サーバがその実行中に実行プランを変更した回数。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	最適化された要求の数
QueryBypassed	プラン・キャッシュから再利用された要求の数
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 を返す。それ以外の場合は、OFF を返します。
RecoveryUrgency	データベースのリカバリに必要と予測される時間
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
RemoteTrunc	SQL Remote Message Agent 用に最後に確認されたログ・オフセット
RollbackLogPages	ロールバック・ログのページ数

プロパティ	説明
SeparateCheckpointLog	データベースのチェックポイント・ログが SYSTEM DB 領域の最後に保持される場合は、ON を返す。それ以外の場合は、OFF を返します。
SeparateForeignKeys	プライマリ・キーと外部キーが別々に保存される場合は ON を返す。それ以外の場合は、OFF 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るログ・オフセット
TableBitMaps	データベースがテーブル・ビットマップをサポートする場合は、ON を返す。それ以外の場合は、OFF を返します。
TempFileName	データベース・テンポラリ・ファイルのパスを含む名前
TempTablePages	テンポラリ・テーブルで使用されるテンポラリ・ファイル内のページ数
TransactionsSpanLogs	トランザクションが複数のログ・ファイルにまたがることができる場合は、ON を返す。それ以外の場合は、OFF 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
VariableHashSize	B ツリー・インデックス用のハッシュ・サイズを指定できる場合は、ON を返す。それ以外の場合は、OFF 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

第 18 章

Adaptive Server Anywhere の制限

この章の内容

この章では、Adaptive Server 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説明します。

サイズと数の制限

Adaptive Server 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。
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)	• 256 GB (1 KB ページに対して) • 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 95/98、Windows NT、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

項目	制限
最大キャッシュ・サイズ (非 AWE キャッシュ) (Windows NT Advanced Server、Windows 2000 Advanced Server、Windows 2000 Enterprise Server、Windows 2000 Datacenter Server、Windows Server 2003 Enterprise Edition、Windows Server 2003 Datacenter Edition)	2.7 GB
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% ～ 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-Tru64 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

項目	制限
カラム数／テーブル	• ((ページ・サイズ)/4)^2 (32 ビット・サーバに対して) • ((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	ファイル・サイズにより制限
ロー数／テーブル	ファイル・サイズにより制限
テーブル数／データベース	2^32 - 2^20 - 1 = 4 293 918 719
テンポラリ・テーブル数／接続	2^20 = 1 048 576
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 = 4 294 967 295
イベント数／データベース	2^31 - 1 = 2 147 483 647
トリガ数／データベース	2^32 - 1 = 4 294 967 295
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。

索引

記号

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

つ

て

と

な

に

ね

は

ひ

ふ

へ

ほ

ま

み

め

も

ゆ

よ

ら

り

る

れ

ろ