SYBASE iAnywhere

SQL Anywhere 10
変更点とアップグレード

改訂 2007 年 3 月

版権と商標

目次

はじめに

このマニュアルの内容

このマニュアルでは、SQL Anywhere 10 とそれ以前のバージョンに含まれる新機能について説明します。

対象読者

このマニュアルは、SQL Anywhere Studio の以前のバージョンのユーザで、今回のバージョンで新しく追加された事項と変更された事項を知りたい方を対象としています。

SQL Anywhere のマニュアル

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

SQL Anywhere のマニュアル

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

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

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

マニュアルの形式

SQL Anywhere のマニュアルは、次の形式で提供されています。

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

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

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

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

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

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

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

表記の規則

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

SQL 構文の表記規則

SQL 構文の表記には、次の規則が適用されます。

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

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

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

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

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

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

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

- **オプション部分**　文のオプション部分は角カッコで囲みます。

  RELEASE SAVEPOINT [*savepoint-name*]

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

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

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

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

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

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

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

オペレーティング・システムの表記規則

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

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

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

ファイルの命名規則

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

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

  ```
  MobiLink¥redirector
  ```

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

  ```
  MobiLink/redirector
  ```

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

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

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

オペレーティング・システム別の *install-dir* のデフォルト・ロケーションの詳細については、「SQLANY10 環境変数」『SQL Anywhere サーバ - データベース管理』を参照してください。

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

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

  オペレーティング・システム別の *samples-dir* のデフォルト・ロケーションの詳細については、「サンプル・ディレクトリ」『SQL Anywhere サーバ - データベース管理』を参照してください。

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

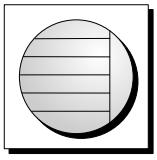
  UNIX、Linux、Mac OS X 環境変数は、*.cshrc* や *.tcshrc* などのシェルとログイン・スタートアップ・ファイルに格納されます。

グラフィック・アイコン

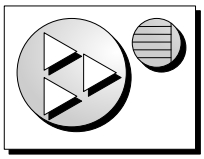
このマニュアルでは、次のアイコンを使用します。

- クライアント・アプリケーション

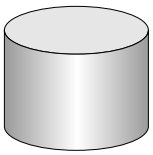
- SQL Anywhere などのデータベース・サーバ



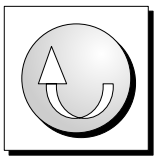
- Ultra Light アプリケーション



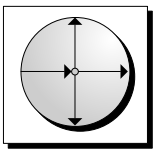
- データベース。高度な図では、データベースとデータベースを管理するデータ・サーバの両方をこのアイコンで表します。



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



- Sybase Replication Server



- プログラミング・インタフェース

詳細情報の検索／フィードバックの提供

詳細情報の検索

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

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

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

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

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

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

フィードバック

このマニュアルに関するご意見、ご提案、フィードバックをお寄せください。

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

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

第 1 章

バージョン 10.0.1 の新機能

目次

SQL Anywhere

Mobile Link

QAnywhere

Ultra Light

SQL Remote

製品全体の機能

廃止される機能は変更される可能性があります
廃止される機能のリストはあくまでも予定であって完全なものとは限らず、変更される可能性があります。

SQL Anywhere

次の項では、SQL Anywhere のバージョン 10.0.1 の新機能、動作の変更、廃止される機能について説明します。

> **注意**
> 10 より前のバージョンでは、SQL Anywhere データベース・サーバは、Adaptive Server Anywhere と呼ばれていました。

新機能

次に、バージョン 10.0.1 で導入された SQL Anywhere データベースとデータベース・サーバの新機能のリストを示します。

暗号化の強化

暗号化のサポートを強化するために、次のような変更が加えられています。

- **CREATE DATABASE 文の ENCRYPTION 句の拡張** CREATE DATABASE 文の ENCRYPTION 句の構文が拡張され、SIMPLE を暗号化タイプとして指定できるようになりました。また、暗号化キーとアルゴリズムを任意の順番で指定できます。「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **dbinit と dbunload の -ea オプションの強化** dbinit と dbunload の -ea オプションでは、暗号化タイプとして、なし (none) と単純 (simple) の両方を使用できるようになりました。none を指定すると、暗号化は行われません。simple を指定すると、単純暗号化が行われます。また、-ea とともに -ek、-et、-ep のどのオプションが指定されたかに応じて、デフォルトの暗号化タイプが変更されました。「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバ - データベース管理』と「アンロード・ユーティリティ (dbunload)」『SQL Anywhere サーバ - データベース管理』を参照してください。

  -e オプションは推奨されなくなりました。「動作の変更と廃止される機能」 9 ページを参照してください。
- **Mac OS X での強力な暗号化** Mac OS X で RSA 暗号化を使用して、クライアント／サーバ通信を暗号化できるようになりました。「SQL Anywhere クライアント／サーバ通信の暗号化」『SQL Anywhere サーバ - データベース管理』を参照してください。

クライアントでの文のキャッシュのサポート

クライアントでの文のキャッシュがサポートされるようになり、デフォルトで有効です。この機能では、同じ SQL 文の準備と削除が繰り返し行われると、クライアントはこの文をキャッシュして、アプリケーションによって文が削除された後もサーバで準備された状態にしておきます。これにより、データベース・サーバは、文の削除や再準備などの余分な作業を節約できます。クライアントでの文のキャッシュを使用するには、バージョン 10.0.1 のクライアント・ライブラリとバージョン 10.0.1 のデータベース・サーバの両方が必要です。

クライアントでの文のキャッシュをサポートするために、次のような変更が加えられています。

- **max_client_statements_cached オプション**　このオプションは、アプリケーションによって文が削除された後であっても、データベース・サーバでキャッシュ (準備) されたまま維持できる文の最大数を指定します。「max_client_statements_cached オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **新しい接続プロパティとサーバ・プロパティ**　ClientStmtCacheHits、ClientStmtCacheMisses、max_client_statements_cached の各プロパティが追加されました。「接続レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』と「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **新しい要求の統計値**　文キャッシュ・ヒットと文キャッシュ・ミスの統計値が追加されました。「要求の統計値」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

SQL FLAGGER の強化

SQL FLAGGER 機能が強化されて、互換性の検出が向上したほか、新しい標準のサポートが追加されました。たとえば、特定の SQL 標準との互換性や Ultra Light SQL との互換性をテストできるようになりました。

これらの強化をサポートするために、次のような変更が加えられています。

- **新しい SQLFLAGGER 関数**　新しい SQLFLAGGER 関数を使用すれば、実際に文を実行しなくても、SQL 文が指定された SQL 標準に準拠しているかどうかをテストできます。「SQLFLAGGER 関数 [その他]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **新しい sa_ansi_standard_packages システム・プロシージャ**　新しい sa_ansi_standard_packages システム・プロシージャを使用して、SQL 標準や SQL 文を指定したり、文の実行中に使用される非コア SQL 拡張の一覧を取得したりすることができます。「sa_ansi_standard_packages システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  この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。「バージョン 10.0.0 のデータベースのアップグレード」 354 ページを参照してください。
- **sql_flagger_error_level データベース・オプションと sql_flagger_warning_level データベース・オプションの新しい値**　sql_flagger_error_level データベース・オプションと sql_flagger_warning_level データベース・オプションで新しい値がいくつか追加され、SQL/1999 標準と SQL/2003 標準がサポートされるようになりました。「sql_flagger_error_level オプション [互換性]」『SQL Anywhere サーバ - データベース管理』と「sql_flagger_warning_level オプション [互換性]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **SQL プリプロセッサ (sqlpp) の -e オプションと -w オプションの新しい値**　SQL プリプロセッサ (sqlpp) の -e オプションと -w オプションで新しい値がいくつか追加され、SQL/1999 標準と SQL/2003 標準がサポートされるようになりました。「SQL プリプロセッサ」『SQL Anywhere サーバ - プログラミング』を参照してください。

SQL 文

SQL 文と関数には、次のように強化が行われています。

- **START DATABASE 文の強化**　START DATABASE 文では、データベースの DB 領域ファイルが配置されているディレクトリを指定できる DIRECTORY 句がサポートされるようになりました。「START DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **INSERT 文、UPDATE 文、DELETE 文、SELECT 文、UNION 文、EXCEPT 文、INTERSECT 文に OPTION 句が含まれる**　INSERT、UPDATE、DELETE、SELECT、UNION、EXCEPT、INTERSECT の各文には OPTION 句があるため、その文が実体化ビュー (Materialized View) を使用する方法とクエリを最適化する方法を指定したり、次のデータベース・オプションについて設定を上書きしたりすることができます。

  - isolation_level
  - max_query_tasks
  - optimization_goal
  - optimization_level
  - optimization_workload

  次の項を参照してください。

  - 「INSERT 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「UPDATE 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「DELETE 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「SELECT 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「UNION 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「EXCEPT 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「INTERSECT 文」『SQL Anywhere サーバ - SQL リファレンス』
- **HTML_DECODE 関数**　HTML_DECODE 関数は、トレードマーク記号 (™) など、数値エンティティとして指定されたユニコード・コードポイントをさらに多く復号化するようになりました。コードポイントがデータベースの文字セットで表すことができない場合は、コードポイント形式のままになります。以前は、0x7F より小さいコードポイントは文字に変換され (一部の文字セットでは 0xFF より小さいコードポイントが文字に変換されていました)、それ以外のコードポイントはすべてコードポイント形式のままでした。「HTML_DECODE 関数 [その他]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

照合の適合化のサポート

SQL Anywhere では、データベースの作成時に照合の適合化がサポートされるようになりました。照合の適合化をサポートするために、次のような変更が加えられています。

- **CREATE DATABASE 文の強化**　CREATE DATABASE 文または初期化ユーティリティ (dbinit) を使用してデータベースを作成するときに、文字のソートや比較を詳細に制御する適合化オプションを指定できるようになりました。

  CREATE DATABASE 文の場合は、COLLATION 句および NCHAR COLLATION 句を使用して照合の適合化がサポートされます。「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

初期化ユーティリティの場合は、-z オプションおよび -zn オプションを使用して照合の適合化がサポートされます。「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバ - データベース管理』を参照してください。

注意
照合の適合化オプションを使用して作成したデータベースは、10.0.1 より前のデータベース・サーバでは起動できません。
既存のデータベースに対して照合の適合化を使用するには、照合の適合化をサポートする新しいバージョン 10.0.1 のデータベースを作成し、既存のデータベースをアンロードして、そのデータベースを新しいバージョン 10.0.1 データベースに再ロードします。「バージョン 10.0.0 のデータベースの再構築」 355 ページを参照してください。

- **新しい HasCollationTailoring データベース・プロパティ** 新しいデータベース・プロパティ HasCollationTailoring は、データベースの作成時に適合化サポートが有効だったかどうかを示します。「データベース・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **新しく拡張されたプロパティ値** Collation、NcharCollation、CatalogCollation の各データベース・プロパティを問い合わせるときに、新しい DB_EXTENDED_PROPERTY 値である CaseSensitivity、AccentSensitivity、PunctuationSensitivity、Properties、Specification を利用できます。「DB_EXTENDED_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **SORTKEY 関数と COMPARE 関数の拡張** 照合名をパラメータとして使用できるだけでなく、SORTKEY 関数と COMPARE 関数では、CREATE DATABASE 文と同じ照合の適合化オプションをカッコで囲んで使用できるようになりました。「SORTKEY 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』と「COMPARE 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

Web サービスの強化

HTTP ヘッダと SOAP ヘッダの設定可能性を改善するために、次のような強化が行われています。

- **設定可能性の改善** CREATE PROCEDURE 文と CREATE FUNCTION 文の新しい SET 句では、HTTP プロトコルや SOAP プロトコルのオプションを修正できます。修正できるオプションは、クライアントが使用する HTTP バージョン、チャンクを使用するかどうか、SOAP 要求で呼び出す SOAP 処理の名前 (プロシージャや関数の名前と異なる場合) です。「CREATE PROCEDURE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **HTTP ヘッダ仕様** CREATE PROCEDURE 文と CREATE FUNCTION 文の HEADER 句の構文が拡張され、指定された HTTP 要求ヘッダを抑制したり、空の値を指定したりすることができるようになりました。この機能は、以前のリリースでは修正できなかった自動的に生成された HTTP 要求ヘッダにも適用されます。「CREATE PROCEDURE 文」『SQL Anywhere サーバ - SQL リファレンス』と「HTTP ヘッダの修正」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **SOAP:RPC クライアントでのデータ型のサポート**　CREATE SERVICE 文の DATATYPE 句を使用して、データ型指定を有効にできます。データ型情報は、すべての SOAP サービス・フォーマットでパラメータ入力と結果セット出力 (応答) の XML エンコードに含まれます。これにより、パラメータを文字列に明示的に変換するクライアント・コードが不要になるため、SOAP ツールキットからのパラメータ受け渡しが簡単になります。「データ型の使用」『SQL Anywhere サーバ - プログラミング』を参照してください。
- **Mac OS X での HTTPS サポート**　以前のリリースでは、Mac OS X では HTTP プロトコルだけがサポートされていました。今回のリリースでは、SQL Anywhere データベース・サーバを Mac OS X 上で Web サーバとして実行しているときに、HTTPS を使用できるようになりました。「-xs サーバ・オプション」『SQL Anywhere サーバ - データベース管理』と「SQL Anywhere Web サービス」『SQL Anywhere サーバ - プログラミング』を参照してください。

データベース・ミラーリングの強化

データベースのミラーリング機能に、次の機能強化が追加されています。

- **データベース・ミラーリングにおける優先データベース・サーバの指定**　ミラーリング・システムでプライマリ・サーバのロールを引き受けるデータベース・サーバを指定できるようになりました。「-xp データベース・オプション」『SQL Anywhere サーバ - データベース管理』と「優先データベース・サーバの指定」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **プライマリ・サーバからミラー・サーバへのデータベース・ミラーリングのフェールオーバの起動**　ALTER DATABASE 文の SET PARTNER FAILOVER 句を使用して、プライマリ・サーバからミラー・サーバへのデータベース・ミラーリングのフェールオーバを起動できるようになりました。「ALTER DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』と「プライマリ・サーバのフェールオーバの起動」『SQL Anywhere サーバ - データベース管理』を参照してください。

SQL Anywhere プラグイン

- **[トリガ] フォルダのカラム名の変更**　[トリガ] フォルダの [テーブル名] カラムと [テーブル所有者] カラムは、[オブジェクト名] カラム、[オブジェクト所有者] カラム、[オブジェクト・タイプ] カラムに置き換えられました。[オブジェクト・タイプ] カラムはデフォルトで表示されませんが、[ビュー] – [カラムの選択] を選択すると表示することができます。
- **ビューのプロパティ・シートに追加された [トリガ] タブ**　非実体化ビュー (Non-materialized View) のプロパティ・シートに、ビューの INSTEAD OF トリガをリストする [トリガ] タブが追加されました。
- **[トリガ作成] ウィザードに追加された INSTEAD OF トリガのサポート**　[トリガ作成] ウィザードでいくつか機能が強化され、INSTEAD OF トリガがサポートされるようになりました。これには、テーブルと非実体化ビュー (Non-materialized View) のどちらに対するトリガを作成するかを選択するオプションの追加も含まれます。「トリガの作成」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **[データベース作成] ウィザードに追加された照合の適合化のサポート**　選択したデータベース・サーバがバージョン 10.0.1 以降のサーバである場合、または新しいデータベース・サーバ

を起動してローカル・コンピュータにデータベースを作成することにした場合、[データベース作成] ウィザードに照合の適合化のページが含まれるようになりました。

その他の機能強化

- **単純な DML 文でのプランのキャッシュ**　プランのキャッシュが拡張され、クエリを省略できる INSERT 文、UPDATE 文、DELETE 文 (単純な文) が含まれるようになりました。「プランのキャッシュ」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **左外部ジョインや右外部ジョインを含む実体化ビュー (Materialized View) がコストベースの最適化中に使用可能**　以前は、左外部ジョインや右外部ジョインは、実体化ビュー (Materialized View) の定義で使用できました。しかし、これが原因で、実体化ビュー (Materialized View) をコストベースの最適化で使用することができませんでした。このバージョンでは、左外部ジョインや右外部ジョインを含む実体化ビュー (Materialized View) がコストベースの最適化中に使用できるようになりました。「実体化ビュー (Materialized View) によるパフォーマンスの向上」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **INSTEAD OF トリガのサポート**　BEFORE トリガまたは AFTER トリガは、それぞれトリガ操作の前または後に起動されます。INSTEAD OF トリガは、トリガ操作を置き換えます。INSTEAD OF トリガを使用することで、挿入、更新、削除の各操作中にトリガの動作をより細かく制御できます。「CREATE TRIGGER 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **DBTools の強化**　DBCreatedVersion 関数を使用することで、データベースが SQL Anywhere 10.0.0 またはそれ以前のバージョンを使用して作成されたかどうかをデータベースを起動せずに判別できるようになりました。「DBCreatedVersion 関数」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **OLAP の強化**　新しい 2 つの Window 集合関数 FIRST_VALUE と LAST_VALUE がサポートされるようになりました。これらの関数は、ウィンドウの最初または最後の値をそれぞれ返すため、セルフジョインを使用してこれらの値を返す必要がありません。ウィンドウ上で実行される以後の計算では、これらの値をベースラインとして使用できます。「FIRST_VALUE 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』と「LAST_VALUE 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **強化された UNIX での IPv6 サポート**　UNIX では、IPv6 アドレスの一部としてインタフェース識別子かインタフェース名を指定できます。Linux (カーネル 2.6.13 以降) では、クライアントまたはサーバで IP アドレスを指定するとき (たとえば HOST=、MYIP=、または BROADCAST= TCP プロトコル・オプションを使用するとき) に、インタフェース識別子が必要です。「SQL Anywhere での IPv6 サポート」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **TDS DATE データ型と TDS TIME データ型のサポート**　TDS DATE データ型と TDS TIME データ型は、最近になって TDS クライアントに導入されました。Open Client 15 やそれより新しいバージョン、または jConnect の EBF を使用するアプリケーションでは、日付カラムや時刻カラムを TDS DATETIME ではなく TDS DATE 値または TDS TIME 値としてフェッチできるようになりました。

  SQL Anywhere では、TDS ベースのアプリケーションで日付や時刻のデータを TDS DATE 値や TDS TIME 値としてフェッチできるように強化されました。Open Client や jConnect の古いバージョンを使用するアプリケーションでは、日付や時刻のデータを引き続き TDS DATETIME

としてフェッチします。TDS ベースでないアプリケーション (Embedded SQL、ODBC、iAnywhere JDBC ドライバを使用するアプリケーション) では、これまでも日付や時刻のデータを日付値や時刻値としてフェッチできていました。

- **使用可能な文字セット・エンコードをリストする新しい dbinit オプション**　データベースで使用可能な文字セット・エンコードをリストするには、初期化ユーティリティ (dbinit) の -le オプションを使用します。「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **新しい -ds サーバ・オプション**　-ds サーバ・オプションを使用して、データベースの DB 領域ファイルが配置されている場所を指定できます。「-ds データベース・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **SADbType.Xml データ型**　SADbType.Xml 列挙定数が SQL Anywhere .NET プロバイダに追加されました。
- **SQL Anywhere SNMP Extension Agent の動的トラップにおける単位のサポート**　動的トラップを設定するとき、k、m、g、t を使用して、トラップの数値をそれぞれキロバイト、メガバイト、ギガバイト、テラバイトの単位で指定できるようになりました。「動的トラップの作成」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **iAnywhere Solutions Oracle ドライバ用 ODBC データ・ソースの作成**　データ・ソース・ユーティリティ (dbdsn) で -or オプションを指定して、iAnywhere Solutions Oracle ドライバ用の ODBC データ・ソースを作成できるようになりました。「データ・ソース・ユーティリティ (dbdsn)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **エラー報告を送信するダイアログの強化**　エラー報告を iAnywhere に送信するかどうかを確認する dbsupport ダイアログに、[エラー報告の表示] ボタンが追加され、エラー報告に含まれる情報を送信前に確認できるようになりました。

動作の変更と廃止される機能

次に、バージョン 10.0.1 で導入された SQL Anywhere データベースとデータベース・サーバに加えられた変更を、カテゴリごとに示します。

動作の変更

- **クエリ内並列処理を使用するときの変更**　クエリ内並列処理は、background_priority が on に設定されている接続には使用されなくなりました。また、現在要求を処理しているサーバ・スレッド数 (ActiveReq サーバ・プロパティ) が、データベース・サーバの使用ライセンスがあるマシンの CPU コア数を最近超えた場合は、クエリ内並列処理は使用されません。「クエリ実行時の並列処理」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

  新しいサーバ・プロパティである ExchangeTasksCompleted は、データベース・サーバが起動された後、クエリ内並列処理に使用された内部タスクの総数を返します。「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **暗号化の結果は決定的ではなくなった**　以前は、ENCRYPT 関数を使用した値の暗号化は決定的でした。2 つの同一な入力文字列と 2 つの同一な暗号化キーを入力すると、同一の出力データ (暗号文) が返されていました。新しい encrypt_aes_random_iv データベース・オプションで

は、暗号化が決定的であるかどうかを制御できるようになりました。新しいデフォルトの動作は、非決定的です。「encrypt_aes_random_iv オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

> **注意**
> このデータベース・オプションのないデータベース・サーバ (バージョン 10.0.0 以前) では、たとえ Off であってもこのオプションが設定されたデータベースからはデータを復号化できません。

- **ALTER DBSPACE RENAME では、DB 領域が開かれていない場合に開こうとする** 以前は、DB 領域を使用するデータベースが起動し、そのうちのいずれかの DB 領域が見つからなかった場合、その DB 領域に対して ALTER DBSPACE ... RENAME 文を実行すると、カタログで DB 領域名が更新されましたが、DB 領域の起動は試みられませんでした。今回のリリースでは、カタログの更新後に、データベース・サーバが DB 領域を開こうとするようになりました。「ALTER DBSPACE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **CREATE DBSPACE 文、ALTER DBSPACE 文、DROP DBSPACE 文に対する変更** CREATE DBSPACE 文と DROP DBSPACE 文では、事前に定義された DB 領域の名前 (SYSTEM、TEMPORARY、TEMP、TRANSLOG、TRANSLOGMIRROR) を受け入れなくなりました。以前のバージョンの SQL Anywhere データベース・サーバで作成されたデータベースのユーザ DB 領域が、事前に定義された DB 領域のいずれかと名前が同じ場合、データベース・サーバは常にユーザ DB 領域を参照します。「事前定義の DB 領域」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **sa_conn_info システム・プロシージャの出力に対する変更** sa_conn_info システム・プロシージャの出力が変更され、接続が待機しているロックに関して詳細な情報が得られるようになりました。LockName フィールドは削除されました。それに代わって、LockRowID と LockIndexID という新しい 2 つのフィールドが追加されました。特定のロー識別子に関連付けられているロックで接続が待機している場合は、LockRowID にそのロー識別子が含まれます。特定のインデックスに関連付けられているロックで接続が待機している場合は、LockIndexID にそのインデックスの識別子が含まれます。「sa_conn_info システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  この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。「バージョン 10.0.0 のデータベースのアップグレード」 354 ページを参照してください。

- **一部のシステム・プロシージャで DBA パーミッションは不要になった** sa_dependent_views、sa_get_dtt、sa_check_commit、sa_materialized_view_info の各システム・プロシージャでは、DBA パーミッションを必要とせず実行できるようになりました。

- **CREATE DATABASE 文のデフォルトの測定単位の削除** CREATE DATABASE 文を使用してデータベースを作成する場合、DATABASE SIZE の値を指定すれば、オプションだった測定単位の指定は不要になりました。「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **default_timestamp_increment オプションの新しい最大値** default_timestamp_increment オプションの最大値が 1000000 (1 秒) になりました。「default_timestamp_increment オプション [デー

タベース] [Mobile Link クライアント]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **dbdata10.dll の削除** dbdata10 DLL (Dynamic Link Library) で提供されていた機能は、SQL Anywhere .NET プロバイダ DLL に組み込まれました。その結果 Windows CE では、SQL Anywhere .NET プロバイダ DLL にプラットフォーム固有のバージョンが用意されました。
- **NetWare におけるデフォルト・キャッシュ・サイズの増加** NetWare におけるデータベース・サーバのデフォルト・キャッシュ・サイズが 2 MB から 8 MB に増加されました。「-c サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **クライアントでの文のキャッシュによる動作の変更** クライアントでの文のキャッシュをサポートした結果、次のような動作の変更が導入されました。
  - 同じ SQL 文で結果セットがないと記述された後になって結果セットが存在するような場合は、不正な記述が発生することがあります。次に例を示します。

    ```
    CREATE PROCEDURE p() NO RESULT SET BEGIN ... END
    Prepare, Describe, Drop "call p"
    ALTER PROCEDURE p() RESULT( ... ) BEGIN ... END
    Prepare, Describe, Drop "call p"       // describe returns no result set
    ```

  - クライアントでの文のキャッシュが有効で、RememberLastStatement が有効な場合 (-zl サーバ・オプション)、キャッシュされた文を再使用するときの LastStatement プロパティは空の文字列になります。
  - クライアントでの文のキャッシュが有効な場合、sa_get_request_times または sa_get_request_profile を使用して要求レベル・ログを処理する場合、文の実行回数が不正である可能性があります。「max_client_statements_cached オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **言語ユーティリティ (dblang) に管理者権限は不要になった** 以前のバージョンの SQL Anywhere では、dblang ユーティリティを使用して SQL Anywhere のローカライズされたバージョンの言語設定を変更するために、ユーザは管理者としてログインする必要がありました。この要件はなくなりました。
- **DISH サービス名にスラッシュを使用できなくなった** DISH サービス名が誤って解釈されないように、スラッシュ (/) をサービス名の一部として使用できなくなりました。「CREATE SERVICE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **SACommand.UpdateRowSource のデフォルト値の変更** これまで SACommand.UpdatedRowSource のデフォルト値は UpdatedRowSource.Both でしたが、UpdatedRowSource.OutputParameters に変更されました。「UpdatedRowSource プロパティ」『SQL Anywhere サーバ - プログラミング』を参照してください。
- **PrefetchRows 接続パラメータのデフォルト値の変更** .NET データ・プロバイダを使用するとき、PrefetchRows 接続パラメータのデフォルト値が 10 から 200 に変更され、パフォーマンスが向上しました。SAConnectionStringBuilder.PrefetchRow のデフォルト値も 200 に変更されました。結果セットに BLOB カラムが含まれる場合、プリフェッチは無効です。「PrefetchRows 接続パラメータ [PROWS]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **認証アプリケーションで saopts.sql ではなく authenticate.sql を使用するようになった**　以前のリリースの SQL Anywhere OEM 版では、データベースの作成、再構築、または更新を行うたびに適用されるように、認証文をファイル *install-dir¥scripts¥saopts.sql* に格納するように推奨されていました。

  今回のリリースでは、認証文字列をファイル *install-dir¥scripts¥authenticate.sql* に格納するように推奨されるようになりました。「認証データベースのアップグレード」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **HP-UX で長いホスト名を使用できるようになった**　HP-UX 11i v2 September 2004 Update より、システム管理者はカーネル・パラメータを設定することで 255 バイトのホスト名のサポートを有効にできるようになりました。しかし、長いホスト名のサポートが有効な HP-UX コンピュータ上の SQL Anywhere サーバで、MachineName プロパティと AppInfo HOST キーは最長で 64 バイトのホスト名を返していました。今回のリリースでは、MachineName と AppInfo の両方で、255 バイトのホスト名を返せるようになりました。

- **iAnywhere JDBC ドライバの URL ヘッダ**　以前のリリースでは、アプリケーションが iAnywhere JDBC ドライバを使用して SQL Anywhere に接続した場合、JDBC ドライバに渡された URL はヘッダ **jdbc:odbc:** で始まっていました。今回のリリースでは、URL のヘッダは **jdbc:ianywhere:** で始めることもできるようになりました。Sun JDBC-ODBC ブリッジとの競合を避けるため、**jdbc:ianywhere:** を使用することをおすすめします。「ドライバへの URL の指定」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **Remarks 値が 128 文字よりも長い場合の jConnect を使用したテーブル・リストの取得**　以前は、JDBC アプリケーションが jConnect を使用して接続し、テーブルのリストを要求した場合、テーブルが存在していたとしても、結果は空になる可能性がありました。string_rtruncation オプションが On に設定され、アプリケーションが DatabaseMetaData.getTables メソッドを使用し、任意のテーブルの Remarks 値が 128 文字より長い場合に、このような状況が発生していました。今回のリリースでは、長すぎる Remarks 値は 128 文字にトランケートされるようになり、テーブルのリストが返されます。この変更を使用するには、*jcatalog.sql* を実行するか、データベースをアップグレードする必要があります。「jConnect システム・オブジェクトのデータベースへのインストール」『SQL Anywhere サーバ - プログラミング』または「アップグレード・ユーティリティ (dbupgrad)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **CHAR 値と NCHAR 値の比較**　SQL Anywhere 10.0.0 では、CHAR ドメインと NCHAR ドメインを組み合わせると、NCHAR の比較になっていました。しかしこれにより、SQL_C_WCHAR としてバインドされたホスト変数を使用すると、10.0.0 にアップグレードしたアプリケーションでは異なる結果が得られたり、パフォーマンスが低下したりする可能性があります。SQL Anywhere 10.0.0 では、SQL_C_WCHAR としてバインドされた変数は NCHAR として表されます。SQL Anywhere 10.0.1 では新しい推定規則が導入されて、既存のアプリケーションとの互換性が向上し、CHAR ドメインと NCHAR ドメインを組み合わせたときに一貫性のある予測可能な結果を提供するようになりました。「CHAR と NCHAR の比較」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **2.6.12 より前の Linux カーネルで非同期 I/O が無効**　2.6.12 より前の Linux カーネルでのバグにより、影響のあるカーネルのいずれかで SQL Anywhere データベース・サーバを実行すると、デフォルトで非同期 I/O が無効になります。非同期 I/O を使用する場合は、カーネルを 2.6.12 以降にアップグレードする必要があります。

- **OEM 版マニュアルの削除**　SQL Anywhere OEM 版の以前のリリースでは、認証アプリケーションを設定するための手順は、*.pdf* または *.html* ファイルに別途記載されていました。今回のリリースでは、この情報は次の場所に記載されています。
  - 「SQL Anywhere の認証アプリケーションの実行」『SQL Anywhere サーバ - データベース管理』
  - 「connection_authentication オプション [データベース]」『SQL Anywhere サーバ - データベース管理』
  - 「database_authentication [データベース]」『SQL Anywhere サーバ - データベース管理』
- **アンロード・ユーティリティの -e と -t オプションで、大文字と小文字が区別されるデータベースにおけるテーブル名の大文字と小文字の区別は不要になった**　以前のリリースでは、dbunload ユーティリティを使用し、-e オプションまたは -t オプションを指定して大文字と小文字が区別されるデータベースをアンロードする場合、これらのオプションには大文字と小文字を区別したテーブル名を指定する必要がありました。今回のリリースでは、テーブル名は大文字と小文字が区別されません。
- **テンポラリ・テーブルへのデータのロード**　テンポラリ・テーブルにデータをロードする際の動作が変更されました。ON COMMIT DELETE ROWS で定義された LOCAL TEMPORARY TABLE を除いて、コミットは、テンポラリ・テーブルで LOAD TABLE を実行する前後に自動的に実行されます。ロードに失敗すると、テンポラリ・テーブル内のすべてのロー (ロードの前に存在したローを含む) が削除されるようになりました。

  ON COMMIT DELETE ROWS で定義された LOCAL TEMPORARY TABLE の場合、動作の変更はなく、コミットは実行されません。つまり、ロード中に発生した障害によって部分的になったロードの場合は、この種のテンポラリ・テーブルには、ロードされたローの一部しか含まれず、ロード以前に存在したローが見つからないこともあります。

  また、別のテーブルの外部キーから参照されるローがテーブルに含まれている場合、テンポラリ・テーブルへのロードは失敗します。
- **UCA 照合を使用した日本語のデータベースに対する大文字と小文字の区別のデフォルト設定**　日本語のデータベースを作成する場合の UCA 照合では、デフォルトで、大文字と小文字およびアクセント記号が*区別されます*。日本語のデータベースとは、OS の言語または文字セットが日本語であるコンピュータで作成されたデータベースや、932JPN や EUC_JAPAN などの日本語の CHAR 照合で作成されたデータベースです。

  日本語以外のデータベースを作成する場合の UCA 照合では、デフォルトで、大文字と小文字が依然として*区別されません*。

  大文字と小文字およびアクセント記号の区別のデフォルト設定は、dbinit の -c と -a (または -c- と -a-) オプションを指定して、それぞれ上書きすることができます。また、照合の適合化構文や、CREATE DATABASE 文の CASE 句と ACCENT 句を使用して上書きすることもできます。「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバ - データベース管理』と「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

非推奨機能とサポートされなくなった機能

- **SQL FLAGGER での SQL/1992 標準サポート**　SQL FLAGGER では、SQL:1992 (全レベル) はサポートされなくなりました。

- **dbinit -e オプションの廃止**　データベースの作成時に単純暗号化を指定するための dbinit -e オプションは廃止されました。単純暗号化を指定するには -ea オプション (実際には -ea simple) を使用してください。「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **SADbType.oldbit データ型の削除**　SADbType.oldbit 列挙定数が SQL Anywhere .NET プロバイダから削除されました。
- **-gx サーバ・オプションの廃止**　Windows デスクトップ・プラットフォームでは、データベース・サーバ・スケジューラが CPU キャッシュを使用できるように要求の類似性の維持を試みるようになりました。その結果、可能なかぎり 1 CPU 上で要求が実行されます。データベース・サーバが使用するオペレーティング・システム・スレッドの数を指定するために使用される -gx サーバ・オプションも同様に廃止されました。データベース・サーバではこのオプションは無視されます。
- **CREATE DATABASE 文の CASE 句と ACCENT 句の廃止**　CREATE DATABASE 文の COLLATION 句および NCHAR COLLATION 句を使用して照合を適合化できるようになったので、CREATE DATABASE 文の CASE 句と ACCENT 句は廃止されました。「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

Mobile Link

次の項では、Mobile Link のバージョン 10.0.1 での新機能、動作の変更、廃止される機能について説明します。

新機能

Mobile Link サーバ

- **Oracle 用の新しい ODBC ドライバ** iAnywhere Solutions 10 - Oracle という名前の Oracle 用の ODBC ドライバが含まれるようになりました。このドライバは、Mobile Link アプリケーション用にカスタマイズされています。

  「Oracle 用の新しい ODBC ドライバ」 24 ページを参照してください。

- **再構築された暗号化レイヤ** Mobile Link の暗号化レイヤが再構築されて改善されました。この変更は透過的であるため、アプリケーションに変更を加える必要はありません。

- **Mobile Link サーバ・ログ・ファイル・ビューワ** 新しいダイアログが追加され、Mobile Link のログ・ファイルを表示することができるようになりました。ログ・ファイル・ビューワでは、ログに記録された情報をフィルタしたり概要や統計値を表示したりといった、強化された機能が提供されます。

  「Mobile Link サーバ・ログ・ファイル・ビューワ」 『Mobile Link - サーバ管理』を参照してください。

- **メモリの使用量の向上** Windows では、必要に応じて、Mobile Link サーバ (32 ビット・プロセス) がより多くのメモリを使用するようになりました。以前のバージョンでは、2 GB 未満に制限されていましたが、必要な場合に使用可能なメモリがあれば、今までよりかなり多いメモリを使用できるようになりました。サーバのメモリ・キャッシュの最大サイズを設定するには、mlsrv10 -cm オプションを使用します。メモリを多く使用した場合は、パフォーマンスが向上するように、ディスクへのページングが減ります。

Sybase Central の Mobile Link プラグイン

- **モデル・モードでの新機能**
  - **テーブル・マッピング** テーブルとカラムのマッピングを作成および変更するのが簡単になりました。テーブルの [マッピング方向] カラムのドロップダウン・リストから選択することで、テーブル・マッピングを有効にできます。同様に、[カラム・マッピング] タブで、カラムをマッピングするかどうかを指定できるようになりました。[テーブル・マッピング新規作成] ウィザードは削除されました。

    「テーブル・マッピングとカラム・マッピングの変更」 『Mobile Link - クイック・スタート』を参照してください。

  - **新しいリモート・テーブルの作成** 統合データベース・スキーマに基づいて新しいリモート・テーブルを作成するのが簡単になりました。[リモート・テーブルの新規作成] ダイアログを使用して、統合データベースのテーブルと同じ名前とカラムで新しいリモート・テーブ

ルを作成できます。このダイアログを使用して、リモート・データベース・テーブルをモデルの統合テーブルにマッピングすることもできます。

「モデルで作成するリモート・データベースの変更」『Mobile Link - クイック・スタート』を参照してください。

- **リモート・データベースのテーブルとカラムの削除** [マッピング] タブで、リモート・データベースのテーブルとカラムを削除できるようになりました。テーブルまたはカラムを選択し、[編集] – [削除] を選択することで、モデルのリモート・データベース・スキーマからリモート・テーブルまたはカラムを削除できます。

  「モデルで作成するリモート・データベースの変更」『Mobile Link - クイック・スタート』を参照してください。

Mobile Link クライアント

- **簡単になった Windows CE 上のネットワーク名の指定方法** default_internet や default_work キーワードを network_name プロトコル・オプションで名前として指定することで、デフォルト設定が使用される名前になるようにすることができるようになりました。

  「network_name」『Mobile Link - クライアント管理』を参照してください。

- **delete_old_logs の機能の強化** データベース・オプション delete_old_logs で、日数を指定できるようになりました。作成されたログは、この期間を過ぎると削除されます。

  「delete_old_logs オプション [Mobile Link クライアント] [SQL Remote] [Replication Agent]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しいフック** 新しいフック sp_hook_dbmlsync_set_ml_connect_info を使用すると、dbmlsync が Mobile Link サーバへの接続を試みる直前に、ネットワーク・プロトコルとネットワーク・プロトコル・オプションを設定することができます。

  「sp_hook_dbmlsync_set_ml_connect_info」『Mobile Link - クライアント管理』を参照してください。

セキュリティ

トランスポート・レイヤ・セキュリティを管理するための、createcert と viewcert という名前の 2 つの新しいユーティリティが追加されました。次の項を参照してください。

- 「トランスポート・レイヤ・セキュリティ」 24 ページ
- 「証明書ユーティリティ」『SQL Anywhere サーバ - データベース管理』

動作の変更と廃止される機能

次に、バージョン 10.0.1 で導入された Mobile Link の変更を示します。

Sybase Central の Mobile Link プラグイン

- **[テーブル・マッピング追加] ウィザードの削除** リモート・データベース・スキーマに対して新しいテーブルを追加するための新機能がモデル・ モードに追加されました。

「テーブル・マッピングとカラム・マッピングの変更」『Mobile Link - クイック・スタート』を参照してください。

- **デフォルトの dbmlsync バッチ・ファイルの改善** Mobile Link モデルを展開するときは、*model-name_dbmlsync.bat* という名前のファイルを作成します。以前のバージョンでは、このファイルのデフォルトの dbmlsync コマンドに -qc オプションが含まれており、このオプションによって同期後に dbmlsync ウィンドウが閉じられていました。そのため、同期が成功したかどうかを判断するのが難しくなっていました。今回のバージョンでは、デフォルトの dbmlsync コマンド・ラインから -qc オプションが削除されました。

セキュリティ

TLS ユーティリティの readcert、gencert、reqtool は廃止されました。createcert と viewcert という名前のユーティリティに置き換えられました。次の項を参照してください。

- 「証明書ユーティリティ」『SQL Anywhere サーバ - データベース管理』

QAnywhere

次の項では、QAnywhere のバージョン 10.0.1 での新機能、動作の変更、廃止される機能について説明します。

新機能

次に、バージョン 10.0.1 で導入された QAnywhere の追加機能を示します。

- **動的アドレス指定** QAnywhere Agent は、有効なネットワークを検出して、再起動しなくても Mobile Link サーバの通信プロトコルとアドレスを自動的に調整できるようになりました。

  「-xd オプション」 『QAnywhere』を参照してください。

- **最大ダウンロード・サイズ** メッセージをダウンロードする最大サイズを設定できるようになりました。

  「-idl オプション」 『QAnywhere』と、「事前に定義されたクライアント・メッセージ・ストア・プロパティ」 『QAnywhere』の ias_MaxDownloadSize を参照してください。

- **QAnywhere サーバ・ログ・ファイル・ビューワ** 新しいビューワが追加され、QAnywhere サーバのログ・ファイルを表示することができるようになりました。ログ・ファイル・ビューワでは、ログに記録された情報をフィルタしたり概要や統計値を表示したりといった、強化された機能が提供されます。

  「QAnywhere サーバのログ」 『QAnywhere』を参照してください。

クライアント API の強化

- **.NET API でのメッセージ・リスナ処理中の例外処理機能** ExceptionListener 委任が .NET API に追加されました。この機能は Java API にすでに存在します。

  次の項を参照してください。

  - 「ExceptionListener 委任」 『QAnywhere』
  - 「ExceptionListener2 委任」 『QAnywhere』

- **メッセージの所有 QAManager をリスナに渡す機能** QAManagerBase API をリスナ内部から呼び出すのに便利な新しいインタフェースが、.NET と Java の API に追加されました。これは、たとえばメッセージの受信確認時に役立ちます。新しいインタフェースでは、QAManagerBase のグローバル・インスタンスを参照したり、QAManagerBase をリスナに渡すために別のコーディング方法を使用したりする必要がありません。

  次の項を参照してください。

  - .NET API：「MessageListener2 委任」 『QAnywhere』
  - Java API：「QAMessageListener2 インタフェース」 『QAnywhere』

動作の変更と廃止される機能

次に、バージョン 10.0.1 で導入された QAnywhere の変更を示します。

- **クライアント・メッセージ・ストアのトランザクション・ログ**　デフォルトでは、クライアント・メッセージ・ストアのトランザクション・ログの内容は、チェックポイントで削除されるようになりました。

  「-m データベース・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

  qaagent -c オプションで StartLine パラメータを指定すると、この動作を変更できます。

  「-c オプション」『QAnywhere』を参照してください。

SQL Remote

次の項では、SQL Remote のバージョン 10.0.1 での動作の変更と廃止される機能について説明します。

動作の変更と廃止される機能

次に、バージョン 10.0.1 で導入された SQL Remote の変更を示します。

- **VIM と MAPI の廃止**　VIM および MAPI メッセージ・システムのサポートは、このリリースで廃止されました。file、ftp、SMTP のメッセージ・タイプは引き続きサポートされます。「メッセージ・タイプの使用」『SQL Remote』を参照してください。

Ultra Light

次の項では、Ultra Light のバージョン 10.0.1 での新機能、動作の変更、廃止される機能について説明します。

新機能

次に、バージョン 10.0.1 で導入された Ultra Light の追加機能を示します。

- **新しいプラットフォームとデバイス**　このリリースでは Windows Vista がサポートされています。
- **SQL パフォーマンスの改善**　これまで Ultra Light では、クエリに既存のインデックスを使用することに利点がないと Ultra Light クエリ・オプティマイザが判断した場合に、プライマリ・キー・インデックスがデフォルトで使用されていました。

  このバージョンでは、プライマリ・キーを使用するのではなく、データベース・ページから直接ローにアクセスします。これによりローはプライマリ・キー・インデックスの順でなくなるため、これまでのリリースとは異なる順序でクエリの結果が返されることがあります。データの順序が重要である場合、クエリで ORDER BY 句を使用してください。「ダイレクト・ページ・スキャンの使用」『Ultra Light - データベース管理とリファレンス』と「インデックス・スキャンの使用」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **SQL によるデータベース・プロパティとオプションのアクセシビリティ**　以前のバージョンでは、各 Ultra Light API のメソッドを使用した場合だけ、データベース・プロパティやオプションにアクセスすることが可能でした。今回のバージョンでは、Ultra Light SQL に次の文と関数が導入されたため、SQL (Interactive SQL を含む) を使用してプロパティやオプションを設定したり取得したりすることができるようになりました。
  - SET OPTION 文。「Ultra Light SET OPTION 文」『Ultra Light - データベース管理とリファレンス』を参照してください。
  - DB_PROPERTY 関数。「DB_PROPERTY 関数 [システム]」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **同時接続数の増加**　Ultra Light では、より多くの同時接続数をサポートするようになりました。データベース 1 つあたりの接続数は 14 のままです。しかし、全体の同時データベース接続数は次のように増えました。
  - Palm OS と Symbian OS の場合はデータベース数が 8、同時接続数が 16
  - その他のすべてのプラットフォームの場合はデータベース数が 32、同時接続数が 64

SQL CA の制限で接続数をより制限可能
エンジンへの接続に使用できる SQLCA の合計数は 31 です。データベース・マネージャあたり 1 つの SQLCA と 接続あたり 1 つの SQLCA を使用するコンポーネントがあることをふまえて、この値を使用してください。つまり、Ultra Light.NET API を使用するアプリケーションでは、実際の接続制限は合計で 30 に減ってしまいます。

- **コミット・フラッシュ操作**　これまでは、COMMIT 文または API 呼び出しによって実行されたコミット操作は、Ultra Light がトランザクションを安全に記憶領域にフラッシュした後でのみ完了していました。このリリースでは、これらの動作を設定したり、独立した操作として論理的に分けたりすることができるようになりました。
  - 論理コミットでは、アプリケーションでトランザクションをロール・バックできるようになりました。
  - チェックポイントでは、エラー後のリカバリ・ポイントを使用できるようになりました。これにより、メモリにフラッシュ済みの最後にコミットされたトランザクションまでリカバリすることができます。

  ただし、オートコミット機能を使用する Ultra Light アプリケーションのパフォーマンスを強化することもできます。特にグループ・コミット・フラッシュを使用する場合に有効です。「Ultra Light でのバックアップとリカバリ」『Ultra Light - データベース管理とリファレンス』と「単一のトランザクションまたはグループ化されたトランザクションのフラッシュ」『Ultra Light - データベース管理とリファレンス』を参照してください。

  この新しい動作は、次の機能でサポートされます。
  - CHECKPOINT 文。「Ultra Light CHECKPOINT 文」『Ultra Light - データベース管理とリファレンス』を参照してください。

    Ultra Light Embedded SQL API と C++ API コンポーネントでの Checkpoint メソッド。その他の言語では、代わりに CHECKPOINT 文を使用する必要があります。次の項を参照してください。
    - Ultra Light Embedded SQL：「ULCheckpoint 関数」『Ultra Light - C/C++ プログラミング』
    - Ultra Light C++：「Checkpoint 関数」『Ultra Light - C/C++ プログラミング』
  - COMMIT_FLUSH 接続パラメータ。「Ultra Light COMMIT_FLUSH 接続パラメータ」『Ultra Light - データベース管理とリファレンス』を参照してください。
  - commit_flush_timeout データベース・オプションと commit_flush_count データベース・オプション。「Ultra Light commit_flush_timeout オプション [テンポラリ]」『Ultra Light - データベース管理とリファレンス』と「Ultra Light commit_flush_count オプション [テンポラリ]」『Ultra Light - データベース管理とリファレンス』を参照してください。両方のオプションの概要については、「Ultra Light でのコミット・フラッシュの考慮事項」『Ultra Light - データベース管理とリファレンス』を参照してください。

動作の変更と廃止される機能

次に、バージョン 10.0.1 で導入された Ultra Light の変更を示します。

- **ulafreg**　ulafreg は変更され、標準出力は利用できなくなりました。このユーティリティは、これまでと同様に、必要なオプションを指定してコマンド・ラインから実行します。しかし、クリップボードに出力をコピーするには、これからは [編集] – [コピー] をクリックする必要があります。任意のテキスト・エディタを使用して、これをファイルに保存できます。

「Ultra Light AppForge レジストリ・ユーティリティ (ulafreg)」『Ultra Light - データベース管理とリファレンス』を参照してください。

- **Windows CE での FIPS サポート**　Windows CE デバイスで FIPS をサポートするために、実行プログラム *install-path¥ultralite¥ce¥arm¥fips¥setup.exe* を実行する必要がなくなりました。これからは、*install-dir¥ultralite¥ce¥arm¥sbgse2.dll* を展開するだけで済みます。

製品全体の機能

次の項では、SQL Anywhere バージョン 10.0.1 のすべてのコンポーネントに影響する新機能、動作の変更、廃止される機能について説明します。

新機能

次に、バージョン 10.0.1 で導入された製品全体の追加機能を示します。

Oracle 用の新しい ODBC ドライバ

iAnywhere Solutions 10 - Oracle と呼ばれる Oracle 用のネイティブ ODBC ドライバが用意されました。

以前のバージョンの iAnywhere では、サード・パーティ製の Oracle ドライバをリブランドしていました。今回のバージョンでは、iAnywhere には SQL Anywhere アプリケーションで使用するための独自の Oracle ODBC ドライバが用意されています。このドライバを使用すると、バグ・フィックスがすばやく行えるようになり、国際文字データの処理機能が改善されます。Oracle 統合データベースを使用して Mobile Link を配備する場合や OMNI を使用して Oracle に接続する場合は、この新しいドライバに切り替えることを強くおすすめします。

「iAnywhere Solutions Oracle ドライバ」 『Mobile Link - サーバ管理』を参照してください。

トランスポート・レイヤ・セキュリティ

- **新しい証明書ユーティリティ** 新しい 2 つのユーティリティ createcert と viewcert を使用すると、セキュリティ証明書を作成、修正、表示できます。これまでは、この目的で gencert、reqtool、readcert の各ユーティリティを使用していましたが、これらのユーティリティは廃止されました。

  viewcert を使用すると、いくつかの種類の PKI オブジェクトを表示できます。これまでは証明書を表示することしかできませんでした。viewcert では、PEM オブジェクトと DER オブジェクトの両方を表示することもできます。これまでは PEM オブジェクトだけを表示することしかできませんでした。viewcert を使用して PEM と DER の間で変換したり、パスワードの暗号化や復号化を実行したりすることもできます。

  createcert は、以前の gencert ユーティリティと reqtool ユーティリティの機能を組み合わせ、さらに新しい機能を追加したユーティリティです。ECC 曲線を作成する場合、これまでは sect163k1 を使用する必要がありましたが、曲線を選択できるようになりました。使用できるキーのサイズは、512 ー 2048 ビットの制限がありましたが、512 ー 16384 ビットになりました。GUID シリアル番号は、デフォルトはありませんでしたが、デフォルトが用意されました。オプションで、別の証明書に署名できる証明書を作成することもできるようになりました。また、証明書のプライベート・キーの使用方法を決定する詳細なオプションを指定できます。最後に、すべてのプライベート・キーにはパスワードが必要でしたが、暗号化されていないプライベート・キーを使用できるようになりました。

  「証明書作成ユーティリティ [createcert]」 『SQL Anywhere サーバ - データベース管理』と「証明書ビューワ・ユーティリティ [viewcert]」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Windows CE での FIPS サポート用にアップグレードされた Certicom Security Builder** SQL Anywhere 製品では、2 つの FIPS 承認モジュールのうちの 1 つを使用できますが、どちらも Certicom 製です。
  - Palm OS の場合、これまでと同様に Security Builder Government Services Edition v1.0.1 を使用する必要があります。
  - Windows CE では、Security Builder Government Services Edition v2.0.0 を使用する必要があります。これらのライブラリは、Windows Mobile で使用するように署名できるためです。

  「FIPS 承認の暗号化テクノロジ」 『SQL Anywhere サーバ - データベース管理』を参照してください。

動作の変更

次に、バージョン 10.0.1 で導入された製品全体の変更を示します。

トランスポート・レイヤ・セキュリティ

- **gencert、readcert、reqtool の廃止** セキュリティ・ユーティリティの gencert、reqtool、readcert は廃止されました。gencert と reqtool は createcert で置き換えられました。readcert は viewcert で置き換えられました。

  「証明書ユーティリティ」 『SQL Anywhere サーバ - データベース管理』を参照してください。

ライセンス

- **.lic ファイルに格納されるようになったサーバのライセンス情報** 以前のリリースでは、SQL Anywhere パーソナル・データベース・サーバ、SQL Anywhere ネットワーク・データベース・サーバ、Mobile Link サーバのライセンス情報は、サーバの実行プログラムに格納されていました。この情報が *.lic* ファイルに格納されるようになりました。このファイルは、サーバの実行プログラムと同じディレクトリにあります。実行プログラムで *.lic* が見つからないと、プログラムは起動しません。

  この変更によって、a_dblic_info 構造体の exename メンバが実行プログラムまたはライセンス・ファイル名を指定できるようになりました。

  次の項を参照してください。

  - 「サーバ・ライセンス取得ユーティリティ (dblic)」 『SQL Anywhere サーバ - データベース管理』
  - 「データベース・サーバの配備」 『SQL Anywhere サーバ - プログラミング』
  - 「Mobile Link サーバの配備」 『Mobile Link - サーバ管理』
  - 「a_dblic_info 構造体」 『SQL Anywhere サーバ - プログラミング』

Windows Vista サポートの問題

SQL Anywhere バージョン 10.0.1 では、Windows Vista オペレーティング・システムをサポートします。Vista で SQL Anywhere ソフトウェアを実行する場合の問題について、次に説明します。

- **Windows Vista のセキュリティ**　Windows Vista には、新しいセキュリティ・モデルが組み込まれています。ユーザ・アカウント制御 (UAC) はデフォルトで有効に設定され、ファイルに書き込み可能とされるプログラムの動作に影響する可能性があります (特に、コンピュータが複数のユーザをサポートしている場合)。ファイルとディレクトリを作成した場所と作成方法によって、あるユーザが作成したファイルを、他のユーザが読み込んだり、書き込むことが許可されなくなる場合があります。SQL Anywhere をデフォルトのディレクトリにインストールした場合は、複数のユーザに読み込み／書き込みアクセスを許可する必要があるファイルおよびディレクトリが適切に設定されます。

- **SQL Anywhere 昇格操作エージェント**　Vista では、ユーザ・アカウント制御がアクティブな状態で実行する場合に、特定のアクションで権限の昇格が必要になります。SQLAnywhere では、次のプログラムで昇格が必要になることがあります。*dbdsn.exe*、*dbelevate10.exe*、*dblic.exe*、*dbsvc.exe*、*installULNet.exe*、*mlasinst.exe*、*ulafreg.exe*。

  次の DLL では、登録または登録の解除時に昇格が必要です。*dbcond10.dll*、*dbctrs10.dll*、*dbodbc10.dll*、*dboledb10.dll*、*dboledba10.dll*。

  ユーザ・アカウント制御がアクティブな Vista システムでは、SQL Anywhere の昇格操作エージェントに対して昇格を確認するメッセージが表示されることがあります。このメッセージは、識別されたプログラムの実行を継続するかどうかを確認したり (管理者としてログオンしている場合)、管理者のクレデンシャルを提供するように求める (管理者以外でログオンしている場合) ため、Vista のユーザ・アカウント制御システムによって発行されるものです。

- **配備環境の変更**　プログラム *dbelevate10.exe* は、昇格された権限が必要な操作を実行するために SQL Anywhere コンポーネントによって内部的に使用されます。この実行プログラムは、SQL Anywhere の配備環境に含まれている必要があります。

- **ActiveSync の削除**　Microsoft ActiveSync ユーティリティは、Vista ではサポートされていません。これは、Windows Mobile Device Center で置き換えられました。SQL Anywhere ActiveSync プロバイダ・インストール・ユーティリティを Windows Mobile Device Center で使用できます。

- **署名された SQL Anywhere 実行プログラム**　Vista での SQL Anywhere 実行プログラムは、iAnywhere Solutions, Inc. によって署名されています。

- **新しいライセンス・ファイル**　10.0.1 のインストールには、SQL Anywhere 用の新しいライセンス・ファイルを作成する手順が含まれます。既存のインストールからのライセンス情報は、実行ファイル内の以前置かれていた場所から抽出され、新しい場所 (実行プログラムと同じディレクトリにある *dbsrv10.lic*、*dbeng10.lic*、*mlsrv10.lic* ファイル) に移動されます。

  「サーバ・ライセンス取得ユーティリティ (dblic)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **サンプル**　サンプルで、1 つ以上のスペースを含む SQL Anywhere インストール・パス名が正しく処理されるようになりました。

- **Windows サービス**　Vista に準拠したサービスでは、デスクトップとの対話が許可されていません。Windows Vista では、(サービス定義で ［デスクトップとの対話をサービスに許可］ が有効になっている場合でも) SQL Anywhere サービスはデスクトップと対話しません。SQL Anywhere データベース・サーバは、dbconsole ユーティリティを使用するか Sybase Central からモニタリングできます。

Sybase Central を Windows Vista で実行している場合、サービスがデスクトップと対話できるオプションは無効になります。

- **AWE キャッシュの使用**　Windows Vista で AWE キャッシュを使用するには、データベース・サーバを管理者として実行する必要があります。AWE キャッシュを使用して昇格していないデータベース・サーバを起動すると、AWE を使用できるように管理者としてデータベース・サーバを実行するように要求する警告が表示されます。「-cw サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **PowerDesigner、InfoMaker、DataWindow.NET**　SQL Anywhere に含まれる PowerDesigner、InfoMaker、DataWindow.NET の各コンポーネントは、Windows Vista では正式にサポートされていません。そのため、Vista 環境でこれらのコンポーネントを実行すると、問題が発生する可能性があります。Vista 環境での実行方法、およびこれらの製品の Vista 対応版の入手方法については、それぞれの製品マニュアルを参照してください。

第 2 章

バージョン 10.0.0 の新機能

目次

SQL Anywhere

- 「新機能」 31 ページ
- 「動作の変更」 65 ページ
- 「非推奨機能とサポートされなくなった機能」 91 ページ

Mobile Link

- 「新機能」 99 ページ
- 「動作の変更と廃止される機能」 112 ページ

QAnywhere

- 「新機能」 122 ページ
- 「動作の変更と廃止される機能」 125 ページ

SQL Remote

- 「新機能」 128 ページ
- 「動作の変更と廃止される機能」 128 ページ

Ultra Light

- 「新機能」 130 ページ
- 「動作の変更と廃止される機能」 142 ページ

Sybase Central と Interactive SQL

- 「新機能」 145 ページ
- 「動作の変更と廃止される機能」 148 ページ

マニュアルの強化

- 「マニュアルの強化」 151 ページ

廃止される機能は変更される可能性があります
廃止される機能のリストはあくまでも予定であって完全なものとは限らず、変更される可能性があります。

SQL Anywhere

次の項では、SQL Anywhere のバージョン 10.0.0 の新機能、動作の変更、廃止される機能について説明します。

注意
10 より前のバージョンでは、SQL Anywhere データベース・サーバは、Adaptive Server Anywhere と呼ばれていました。

新機能

次に、バージョン 10.0.0 で導入された SQL Anywhere データベースとデータベース・サーバの新機能のリストを示します。

主な機能

- **パフォーマンスを向上させる並列処理のサポート** データベース・サーバで、単一のクエリを処理するのに複数のプロセッサを使用できるようになりました。クエリ内並列処理は、同時実行されるクエリ数が使用可能なプロセッサ数よりも少ない場合に便利です。「クエリ実行時の並列処理」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **データベースのミラーリングのサポート** SQL Anywhere で、データベースの可用性を高めるためのメカニズムである、データベースのミラーリングがサポートされるようになりました。データベースのミラーリングでは、別々のコンピュータで実行され、同期モードか非同期モードで相互通信する 2 ー 3 のデータベース・サーバを使用します。「データベース・ミラーリングの概要」『SQL Anywhere サーバ - データベース管理』を参照してください。

  データベースのミラーリングをサポートするために、次の機能が追加されています。

  - 「synchronize_mirror_on_commit オプション [データベース]」『SQL Anywhere サーバ - データベース管理』
  - データベース・サーバの代替サーバ名。「-sn オプション」『SQL Anywhere サーバ - データベース管理』と「START DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - ServerName プロパティ
  - AlternateServerName プロパティ
  - RetryConnectionTimeout プロパティ
  - ALTER DATABASE *dbname* FORCE START。「ALTER DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - MirrorServerDisconnect システム・イベントと MirrorFailover システム・イベント。「システム・イベントの概要」『SQL Anywhere サーバ - データベース管理』を参照してください。

- 「-xf サーバ・オプション」 『SQL Anywhere サーバ - データベース管理』
- 「-xp データベース・オプション」 『SQL Anywhere サーバ - データベース管理』
- SQL Anywhere SNMP Extension Agent 用の新しいトラップ。「トラップの使用」 『SQL Anywhere サーバ - データベース管理』を参照してください。
- EVENT_PARAMETER 関数に追加された MirrorServerName パラメータ。「EVENT_PARAMETER 関数 [システム]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

データベースのミラーリングに加え、SQL Anywhere では、Veritas Cluster Server エージェントがデータベース用 (SADatabase エージェント) およびデータベース・サーバ用 (SAServer エージェント) に提供されるようになりました。「SQL Anywhere Veritas Cluster Server エージェントの使用」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **スナップショット・アイソレーションのサポート** スナップショット・アイソレーションを使用すると、ユーザがデータを変更する間、データベースでは元のデータのコピーが保持されるので、他のユーザも元のデータを読むことができます。スナップショット・アイソレーションはユーザに対して完全に透過的であり、デッドロックやロック競合の発生を抑えるのに役立ちます。「スナップショット・アイソレーション」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

  スナップショット・アイソレーションをサポートするために、次の機能が追加または強化されました。

  - 「allow_snapshot_isolation オプション [データベース]」 『SQL Anywhere サーバ - データベース管理』
  - 「isolation_level オプション [互換性]」 『SQL Anywhere サーバ - データベース管理』
  - 「sa_snapshots システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
  - 「sa_transactions システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
  - LockCount、SnapshotCount、SnapshotIsolationState、VersionStorePages の各データベース・プロパティ
  - allow_snapshot_isolation、LockCount、SnapshotCount の各接続プロパティ
  - バージョン・ストア・ページ (パフォーマンス・モニタの統計)
  - 「SET 文 [T-SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
  - 「OPEN 文 [ESQL] [SP]」 『SQL Anywhere サーバ - SQL リファレンス』
  - 「ValuePtr パラメータ」 『SQL Anywhere サーバ - SQL の使用法』

- **アプリケーション・プロファイリングと診断トレーシングのサポート** ストアド・プロシージャ・プロファイリングや要求ロギングなどの既存のアプリケーション・プロファイリング機能は、単一の対話型インタフェースである Sybase Central 用 SQL Anywhere プラグインに統合されました。Sybase Central からアプリケーションをプロファイリングするときは、データベースのパフォーマンスを向上させるためのアドバイスが提供されます。

  Sybase Central でのアプリケーション・プロファイリングの詳細については、「アプリケーション・プロファイリング」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **実体化ビュー (Materialized View) のサポート** データベースのサイズが大きく、頻繁にクエリが行われるために大量のデータで繰り返し集約やジョイン操作が発生するような環境において、パフォーマンスを向上させるために、SQL Anywhere では実体化ビュー (Materialized View)

がサポートされました。「実体化ビュー (Materialized View) の編集」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

データベース・サーバは強化され、クエリの一部に応答するために使用できる実体化ビュー (Materialized View) を自動的に決定できるようになりました。この決定はコストを基に行われ、クエリが直接参照するベース・テーブルを使用する必要はありません。「実体化ビュー (Materialized View) によるパフォーマンスの向上」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

実体化ビュー (Materialized View) の情報を格納するために、2 つの新しいシステム・テーブル ISYSMVOPTION と ISYSMVOPTIONNAME が追加されました。「SYSMVOPTION システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSMVOPTIONNAME システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **NCHAR データのサポート**　SQL Anywhere で NCHAR データ型がサポートされるようになりました。NCHAR データ型は、ユニコード文字データを格納するのに使用されます。「NCHAR データ型」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  NCHAR をサポートするために、次の新しい関数が追加されました。

  - 「UNISTR 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「CONNECTION_EXTENDED_PROPERTY 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「UNICODE 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「NCHAR 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「TO_CHAR 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「TO_NCHAR 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』

  NCHAR データ型をサポートするために、次の関数 SOFTKEY と COMPARE に新しいパラメータが追加されました。

  - 「SORTKEY 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「COMPARE 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』

  ユニコード照合アルゴリズム (UCA) を使用するときに、マルチバイトの文字セットを正しくソートできるようになりました。

  NCHAR データ型をサポートするために、初期化ユーティリティ (dbinit) と Unload (dbunload) ユーティリティにも新しいオプションが追加されました。「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバ - データベース管理』と「アンロード・ユーティリティ (dbunload)」『SQL Anywhere サーバ - データベース管理』を参照してください。

  ユニコードのサポートで、ユニコード用インターナショナル・コンポーネント (ICU) を使用するようになりました。「国際言語と文字セット」『SQL Anywhere サーバ - データベース管理』を参照してください。

  ICU の使用と NCHAR データの処理をサポートするために、次のプロパティが変更されました。

  - 新しいデータベースと接続の拡張プロパティ NcharCharSet が追加されました。このプロパティは、データベースまたは接続で使用されている NCHAR 文字セットを返します。

- 新しいデータベース・プロパティ AccentSensitive が追加されました。このプロパティは、アクセントを区別する機能のステータスを返します。
- CharSet データベースと接続のプロパティが、拡張プロパティになりました。

「データベース・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』と「接続レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **内部パフォーマンスの強化** データベース・サーバのパフォーマンスを向上させるために、仮想マシン・テクノロジを使用して、SQL 式の表現と評価を再構成するようになりました。これにより、スループットが大幅に向上します。
- **ビューの依存性のサポート** ビューの依存性の情報がカタログに格納されるようになりました。カタログは特に、データベースの各ビューが依存するビュー、テーブル、カラムを追跡します。ビューが依存するオブジェクトを変更すると、ビュー定義が不適切な結果を返す状態のままにならないように、データベース・サーバは自動的に追加処理を実行します。「ビューの依存性」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

  システム・オブジェクトとそれらの依存性の情報を格納するために、2 つの新しいシステム・テーブル ISYSDEPENDENCY と ISYSOBJECT が追加されました。「SYSDEPENDENCY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSOBJECT システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **チェックポイント・アルゴリズムの向上** データベース・サーバは、チェックポイントを開始し、チェックポイントの発生中にその他の操作を実行できるようになりました。以前は、チェックポイントが発生すると、すべてのアクティビティが停止していました。チェックポイントがすでに進行中の場合、新しくチェックポイントを開始する ALTER TABLE や CREATE INDEX などの操作は、現在のチェックポイントが完了するまで待機する必要があります。「チェックポイントとチェックポイント・ログ」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **ロックの強化** ロックには、次のような強化が行われています。
  - **ロックのクラス** SQL Anywhere で、スキーマ・ロック、テーブル・ロック、ロー・ロック、位置ロックの 4 つの固有のクラスが使用できるようになりました。ロックの問題をより厳密に分析できるように、各トランザクションが保持するロックの種類を明確に記述するように sa_locks システム・プロシージャが変更されました。「ロックの仕組み」『SQL Anywhere サーバ - SQL の使用法』と「sa_locks システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **意図的ロックのサポート** 新しい種類のロックである意図的ロックが、テーブル・ロックとロー・ロックに導入されました。意図的ロックは、アプリケーションがテーブルまたはそのテーブル内のロー・セットの更新意図を通知するために使用されます。アプリケーションで SELECT FOR UPDATE または FETCH FOR UPDATE 文 (または各種プログラミング・インタフェースで同様の構成) を使用するときに、意図的ロックが取得されるようになりました。意図的ロックは、その他の意図的ロックや書き込みロックをブロックしますが、読み込みロックはブロックしません。このため、明示的な同時制御メカニズムとしてロックを使用するアプリケーションで、高度な同時実行性が実現されます。「カーソルの使い方」『SQL Anywhere サーバ - プログラミング』と「意図的ロック」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **一部の状況におけるキー範囲ロックの省略**　インデックス管理アルゴリズムが変更され、データベース・サーバがキーの範囲ではなく個別のインデックス・エントリに、書き込みロックを設定できるようになりました。これにより、同時実行性が向上し、さまざまな環境で同時に発生する INSERT 操作による不要なブロックを避けることができます。「ロックの仕組み」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **インデックス処理の強化**　インデックス処理には、次のような強化が行われています。

  - **新しいインデックスの実装**　SQL Anywhere の以前のリリースには、2 種類のインデックス処理が実装されていました。これらは宣言されたインデックス・カラムのサイズに基づいて自動的に選択されていました。SQL Anywhere 10 で、圧縮 B ツリーインデックスの新しい実装が全体に使用されるようになり、以前の B ツリー・インデックス処理テクノロジが廃止されました。新しいインデックスには、ローの値とは全く別の、インデックス・エントリのインデックス・キー値が圧縮形式で格納されます。スナップショット・アイソレーションをサポートするにはこの機能が必要です。

  - **スナップショット・アイソレーションのサポート**　以前の SQL Anywhere リリースでは、UPDATE または DELETE 文によりインデックス・エントリがすぐに削除されていました。スナップショット・アイソレーションをサポートするために、異なるインデックス・キー値を持つ同じ論理ローを指すような複数のインデックス・エントリが存在する可能性があります。これらの複数のインデックス・エントリは、データベース・サーバによって管理されるため、どの接続でも、任意のローに対してエントリの 1 つしか確認できません。サーバ内のデーモンは、定期的にこれらの余分なインデックス・エントリが (トランザクション COMMIT または ROLLBACK で) 不要になった時点で物理的に削除します。コミットされていない DELETE ではインデックス・エントリを保持することにより、SQL Anywhere の同時実行性制御メカニズムのセマンティック一貫性も向上します。「スナップショット・アイソレーション」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **BLOB 記憶域制御とパフォーマンスの向上**　テーブルのローに (インラインで) 格納される BLOB 値の量を制御できるようになりました。また、BLOB 値をインデックス処理するかどうかも制御できます。これらの強化により BLOB の検索とアクセスが向上します。これらの強化機能を使用するには、CREATE TABLE と ALTER TABLE 文の 3 つの新しい句 INLINE、PREFIX、[NO] INDEX を使用します。BLOB 値は同じテーブルの行同士または行内で共有できるようになりました。重複した BLOB 値を格納する必要がなくなるため、記憶域の要件が抑えられます。「BLOB ストレージ」『SQL Anywhere サーバ - SQL の使用法』、「CREATE TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』、「ALTER TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **カラム圧縮のサポート**　テーブルの個別カラムを圧縮できるようになりました。圧縮には deflate 圧縮アルゴリズムが使用されます。これは、COMPRESS 関数で使用される圧縮方式と同じであり、Windows の *.zip* ファイルで使用されるアルゴリズムです。「CREATE TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』と「ALTER TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **テーブルの暗号化のサポート**　データをセキュリティ保護するためにデータベース全体を暗号化するのではなく、データベースの個別のテーブルを暗号化できるようになりました。テーブルの暗号化は、データベースを初期化するとき有効にする必要があります。「テーブル暗号化」『SQL Anywhere サーバ - データベース管理』を参照してください。

データベース接続

- **一重引用符または二重引用符のサポート**　接続文字列内の値を、一重引用符または二重引用符で囲むことができるようになりました。これによって、接続文字列の値でスペースやセミコロンなどの文字を使用できるようになりました。「接続文字列として渡される接続パラメータ」『SQL Anywhere サーバ - データベース管理』と「接続パラメータのヒント」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **接続文字列でのブール値としての T、Y、F、N の使用**　接続文字列で接続パラメータとプロトコルのオプションを指定するときに、true を示すために T または Y を、false を示すために F または N を指定できるようになりました。「接続パラメータ」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **一部の接続文字列とプロトコルのオプションでの k、m、g サフィックスの付いた値の使用**　次の接続パラメータとプロトコルのオプションで、キロバイト、メガバイト、ギガバイトを表すサフィックスとして k、m、g を使用できるようになりました。
  - 「CommBufferSize 接続パラメータ [CBSIZE]」『SQL Anywhere サーバ - データベース管理』
  - 「CompressionThreshold 接続パラメータ [COMPTH]」『SQL Anywhere サーバ - データベース管理』
  - 「PrefetchBuffer 接続パラメータ [PBUF]」『SQL Anywhere サーバ - データベース管理』
  - 「LogMaxSize プロトコル・オプション [LSIZE]」『SQL Anywhere サーバ - データベース管理』
  - 「MaxRequestSize プロトコル・オプション [MAXSIZE]」『SQL Anywhere サーバ - データベース管理』
  - 「ReceiveBufferSize プロトコル・オプション [RCVBUFSZ]」『SQL Anywhere サーバ - データベース管理』
  - 「SendBufferSize プロトコル・オプション [SNDBUFSZ]」『SQL Anywhere サーバ - データベース管理』
- **AppInfo が Windows クライアントの IP アドレスを返す**　以前のリリースでは、AppInfo 接続パラメータは、UNIX と NetWare クライアントのクライアント・コンピュータの IP アドレスを返すだけでした。Windows クライアントの IP アドレスも返すようになりました。「AppInfo 接続パラメータ [APP]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **個々の接続の監査**　conn_auditing テンポラリ・データベース・オプションがログイン・プロシージャで設定されているときは、特定の接続の監査を有効または無効にできます。データベースの監査ステータスの情報を取得できるように、Auditing データベース・プロパティが追加されました。「conn_auditing オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **RetryConnectionTimeout 接続パラメータ**　RetryConnectionTimeout (RetryConnTO) 接続パラメータは、サーバが見つからない場合に、指定された期間、クライアント・ライブラリに対して接続の試行をリトライするように通知します。「RetryConnectionTimeout 接続パラメータ [RetryConnTO]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **IPv6 のサポート**　IPv6 は Windows、Linux、Mac OS X、Solaris、AIX、HP-UX でサポートされるようになりました。これらのオペレーティング・システムを実行しているサーバで、使用可能なすべての IPv4 と IPv6 のアドレスを受信するようになりました。また、クライアントやサーバで IP アドレスを指定できる箇所で (HOST=、MYIP=、BROADCAST= の各 TCP プロト

コル・オプションなど)、IPv6 アドレスを指定できるようになりました。「SQL Anywhere での IPv6 サポート」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **LDAP 登録の新しいパラメータ** データベース・サーバが Active Directory サーバである場合に、read_authdn と read_password パラメータを使用して、LDAP にデータベース・サーバを登録できるようになりました。「LDAP サーバを使用した接続」『SQL Anywhere サーバ - データベース管理』を参照してください。

バックアップとリカバリ

- **重要なデータベース・ページでのチェックサム自動計算** データベースでチェックサムが有効であるかどうかに関係なく、データベース・サーバは、重要なデータベース・ページのチェックサムを記録します。その結果、データベースでチェックサムが有効でない場合でも、データベースを検証するとチェックサム違反の警告が表示されることがあります。また、破壊された重要なページにアクセスしようとすると、致命的なエラーが発生してデータベース・サーバは停止します。「チェックサムの検証」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **リカバリの開始時に複数のトランザクション・ログを適用** デフォルトでは、データベースのリカバリ時にトランザクション・ログを正しい順序で個々に適用する必要があります。データベース・サーバの開始時に新しいリカバリ・オプション -ad、-ar、-as を指定した場合は、トランザクション・ログをデータベースに適用する順序を手動で指定する必要はありません。データベース・サーバとデータベースはトランザクション・ログの適用中も実行しているため、サーバのキャッシュはウォーミング・ステータスのままになり、総リカバリ時間を減らすことができます。「-ad データベース・オプション」『SQL Anywhere サーバ - データベース管理』、「-ar データベース・オプション」『SQL Anywhere サーバ - データベース管理』、「-as データベース・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **並列データベース・バックアップのサポート** SQL Anywhere データベース・サーバで、サーバ側のイメージ・バックアップで並列バックアップがサポートされるようになりました。並列データベース・バックアップでは、物理 I/O を利用して、直列ではなく並列に情報を読み書きすることで、パフォーマンスが向上します。並列バックアップは次のいずれかの方法で実行できます。
  - 「バックアップ・ユーティリティ (dbbackup)」『SQL Anywhere サーバ - データベース管理』
  - 「BACKUP 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「db_backup 関数」『SQL Anywhere サーバ - プログラミング』
- **最後のバックアップの情報の追跡** 最後のバックアップの情報を格納するために、新しいカラム LAST_BACKUP が ISYSHISTORY システム・テーブルに追加されました。「SYSHISTORY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

セキュリティ

この項では、セキュリティを向上するために SQL Anywhere に加えられた強化について説明します。

- **SQL Anywhere に組み込まれた RSA**　RSA 暗号化を使用するためのライセンスを別に購入する必要がなくなりました。「別途ライセンスが必要なコンポーネント」『SQL Anywhere 10 - 紹介』を参照してください。
- **FIPS サポートの強化**　データベース・サーバに次の FIPS 関連の変更が加えられました。
  - FIPS DLL の名前が *dbrsa10f.dll* から *dbfips10.dll* に変更されました。
  - HASH 関数で 2 つの新しいアルゴリズム SHA1_FIPS と SHA256_FIPS が使用できるようになりました。これらは SHA1 アルゴリズムや SHA256 アルゴリズムと同じですが、FIPS 検証された Certicom バージョンです。
  - -fips サーバ・オプションを指定したときに非 FIPS アルゴリズムを HASH 関数に指定すると、データベース・サーバでは SHA1 の代わりに SHA1_FIPS が、SHA256 の代わりに SHA256_FIPS が使用されます。また、MD5 を使用した場合はエラーが返されます (MD5 は FIPS アルゴリズムではありません)。
  - -fips オプションを指定した場合は、パスワード・ハッシュ処理に SHA256_FIPS が使用されます。

  また、-fips オプションと FIPS 機能をより多くのプラットフォームで使用できるようになりました。-fips オプションをサポートするプラットフォームのリストについては、「サポートされるプラットフォーム」『SQL Anywhere 10 - 紹介』を参照してください。
- **Kerberos 認証**　SQL Anywhere で、Kerberos 認証がサポートされるようになりました。Kerberos 認証では、ユーザ ID やパスワードを指定しなくても、Kerberos クレデンシャルを使用してデータベースに接続できます。「Kerberos 認証の使用」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **新しい権限の追加**　次の権限が追加されました。
  - **BACKUP 権限**　ユーザがバックアップを実行できるようにするために、ユーザ DBA 権限を付与する代わりに、BACKUP 権限をユーザに割り当てることができます。「BACKUP 権限の概要」『SQL Anywhere サーバ - データベース管理』を参照してください。
  - **VALIDATE 権限**　検証操作用の新しい権限 VALIDATE が追加されました。さまざまな VALIDATE 文による操作 (データベース、テーブル、インデックス、チェックサムなどの検証) を実行するには、VALIDATE 権限が必要です。「VALIDATE 権限の概要」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **データベース・サーバのセキュリティ保護機能**　-sf データベース・サーバ・オプションを使用して、データベース・サーバで実行しているデータベースに対してセキュリティ保護されている (無効である) 機能または機能グループを指定できます。「-sf サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

  -sk サーバ・オプションでは、secure_feature_key データベース・オプションを使用するときに、無効になっている機能を有効にするためのキーを指定できます。sa_server_option プロシージャで SecureFeatures プロパティを使用すると、無効になっている機能のセットを変更することもできます。「-sk サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

データベース・ユーティリティ

- **@filename を複数のユーティリティで再使用可能**　コマンド・パラメータ・ファイルを使用するユーティリティで、そのパラメータ・ファイルを個々に解析できるようになりました。パラメータ・ファイル内に配置された簡単な条件ディレクティブを基に解析が行われます。「設定ファイルでの条件付き解析の使用」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **データ・ソース・ユーティリティ (dbdsn) の強化**　dbdsn ユーティリティに次のオプションが追加されています。
  - **-dr**　データ・ソースの作成に使用したコマンドをリストするときは、DRIVER= パラメータを指定します。これによって、データ・ソースを再作成して、現在のバージョンのソフトウェアに含まれる ODBC ドライバとはバージョンの異なる ODBC ドライバを使用できるようになります。
  - **-f**　使用されているシステム情報ファイル (通常は *.odbc.ini*) の名前を表示します。
  - **-ns**　dbdsn に対して、システム情報ファイル (通常 *.odbc.ini*) を検索せずに、既存の環境変数を使用してファイルの場所を判断するように通知します。1 つまたは複数の環境変数によって指定されたファイルが存在せず、ODBC データ・ソースを作成している場合に、この機能が役立ちます。
  - **-pe**　データ・ソースのパスワード・フィールドを暗号化します。

  「データ・ソース・ユーティリティ (dbdsn)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **ヒストグラム・ユーティリティ (dbhist) の強化**　dbhist で作成される Excel 出力ファイル内のシートに対して、Sheet1、Sheet2 などではなく、シートの適用先カラム名を反映した名前が付くようになりました。「ヒストグラム・ユーティリティ (dbhist)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **情報ユーティリティ (dbinfo) の強化**　-u オプションに実体化ビュー (Materialized View) の情報が含まれるようになりました。「情報ユーティリティ (dbinfo)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **初期化ユーティリティ (dbinit) の強化**　初期化ユーティリティ (dbinit) で、次の新しいオプションがサポートされるようになりました。
  - **-a**　UCA 文字列比較でアクセントを区別します。
  - **-af**　UCA 文字列比較でフランス語用のアクセント区別ルールを使用します。
  - **-dba**　新しいデータベースで、デフォルト DBA データベース・ユーザのユーザ ID やパスワードを変更します。
  - **-dbs**　データベース・ファイルの初期サイズを指定します。
  - **-ze**　CHAR データ型の文字セット・エンコーディングを指定します。
  - **-zn**　NCHAR データ型の照合順を指定します。

「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Log Transfer Manager (LTM) の強化** Log Transfer Manager (LTM) ユーティリティ (Replication Agent とも呼ばれる) では、Replication Server 15.0 と Open Server/Open Client 15.0 環境で Replication Agent を使用する場合に、テーブル、カラム、プロシージャ、関数、パラメータの名前に 128 バイトまでの識別子を使用できるようになりました。以前のバージョンでは、識別子は 30 バイトまでに制限されていました。「Replication Server の識別子」『SQL Anywhere サーバ - データベース管理』を参照してください。

  dbltm が生成する情報、警告、エラーの各メッセージのタイムスタンプは、明確に定義された ISO 8601 日時フォーマット (**{I|W|E}** *yyyy-mm-dd hh:mm:ss message*) を使用するようになりました。

- **Ping ユーティリティ (dbping) の強化** Ping ユーティリティ (dbping) で -s または -st オプションを指定すると、Embedded SQL 接続のパフォーマンスやネットワークのパフォーマンスに付いての情報を取得できます。これらのオプションは、dbping を実行するコンピュータと、データベース・サーバを実行するコンピュータとの間のパフォーマンスに関する統計をレポートします。「Embedded SQL 接続のパフォーマンスのテスト」『SQL Anywhere サーバ - データベース管理』を参照してください。

  -pd オプションを使用して、プロパティ値の取得元データベースの名前を指定できるようになりました。「Ping ユーティリティ (dbping)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **サーバ列挙ユーティリティ (dblocate)の強化** サーバ列挙ユーティリティ (dblocate) で、データベースを検索するための新しいオプションがサポートされるようになりました。

  - **-d** サーバの名前とアドレス、および各サーバで実行しているすべてのデータベースのリスト (カンマ区切り) を表示します。
  - **-dn** サーバが指定された名前のデータベースを実行している場合にかぎり、そのサーバの名前とアドレスを表示します。
  - **-dv** サーバの名前とアドレス、および各サーバで実行しているすべてのデータベースのリスト (1 行に 1 つずつ) を表示します。
  - **-p** 指定した TCP/IP ポート番号を使用しているサーバを表示します。
  - **-s** 指定した名前のサーバを表示します。
  - **-ss** 指定したサブ文字列を含むサーバの名前を表示します。

  「サーバ列挙ユーティリティ (dblocate)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **サービス・ユーティリティ (dbsvc) の強化** サービス・ユーティリティ (dbsvc) では、Log Transfer Manager のサービスを管理できる DBLTM サービス・タイプと、Listener ユーティリティのサービスを管理できる dbsln サービス・タイプがサポートされるようになりました。

  サービス・ユーティリティでは、ユーティリティの出力のログをファイルに取ることができる -o オプションも使用できます。「Windows 用サービス・ユーティリティ (dbsvc)」『SQL

Anywhere サーバ - データベース管理』と「Linux 用サービス・ユーティリティ (dbsvc)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しい SQL Anywhere Broadcast Repeater ユーティリティ (dbns10)** SQL Anywhere Broadcast Repeater ユーティリティを使用すると、UDP ブロードキャストは通常到達しない、別のサブネット上やファイアウォール越しの SQL Anywhere データベース・サーバであっても、SQL Anywhere クライアントは HOST パラメータや LDAP を使用することなく検索できます。「SQL Anywhere Broadcast Repeater ユーティリティ (dbns10)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **SQL Anywhere レポート送信ユーティリティ (dbsupport)** 新しい SQL Anywhere レポート送信ユーティリティ (dbsupport) では、エラー・レポートと統計の送信機能、更新 (EBF の可用性) のクエリ機能、以前に送信した問題が修正されたかどうかのチェック機能が提供されます。「SQL Anywhere サポート・ユーティリティ (dbsupport)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **アンロード・ユーティリティ (dbunload) の強化** dbunload には、次のような強化が行われています。
  - dbunload がエラーを検出したときに、未処理の文のログが取られるようになりました。「アンロードの失敗」『SQL Anywhere サーバ - データベース管理』を参照してください。
  - アンロードしたテーブルでバイナリ・データがサポートされるようになりました。
  - データベースのアンロード処理のパフォーマンスを向上するために、さまざまな内部処理が強化されました。

  次の新しいオプションが追加されました。
  - **-dc** データベースのすべての計算カラムの値を再計算します。
  - **-g** 再ロード時に実体化ビュー (Materialized View) を初期化します。
  - **-k** トレース・サポート用に補助テーブルを作成します。このオプションを指定すると、sa_diagnostic_auxiliary_catalog テーブルに移植されます。このオプションは、トレーシング・データベースの作成時に役立ちます。
  - **-nl** 各テーブルに LOAD TABLE 文と INPUT 文を含むがデータはない *reload.sql* ファイルを作成します。

  「アンロード・ユーティリティ (dbunload)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **検証ユーティリティ (dbvalid)** 新しいデータベース検証オプション -d が追加されました。このオプションを指定すると、チェックサム検証、孤立したテーブル・ページと BLOB の検査、構造検査などのデータベース検証を実行します。データはチェック・アウトされません。「検証ユーティリティ (dbvalid)」『SQL Anywhere サーバ - データベース管理』を参照してください。

データベース・オプション

次のデータベース・オプションが追加または強化されました。

- **ansi_substring データベース・オプション**　このオプションは、SUBSTRING 関数の動作を制御します。デフォルトでは、SUBSTRING 関数の動作が ANSI/ISO SQL/99 の動作と一致するようになりました。開始オフセットが負または 0 の場合は、文字列の左側が文字以外で埋められているかのように扱われ、このときに負の長さが指定されるとエラーになります。「ansi_substring オプション [互換性]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **collect_statistics_on_dml_updates データベース・オプション**　DML 文 (INSERT、DELETE、および UPDATE) を実行中の統計の収集を制御します。「collect_statistics_on_dml_updates オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **default_dbspace オプション**　テーブルを作成するデフォルトの dbspace を指定できます。「default_dbspace オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **http_session_timeout オプション**　http_session_timeout オプションを使用して、さまざまなセッション・タイムアウトを制御できます。このオプションは分単位で指定します。デフォルトのパブリック設定は 30 分です。最小は 1 分で、最大は 525600 分 (365 日) です。「http_session_timeout オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **max_temp_space データベース・オプション**　max_temp_space オプションを指定したときに接続で使用可能なテンポラリ領域の最大量を指定できます。「max_temp_space オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **materialized_view_optimization データベース・オプション**　オプティマイザによる実体化ビュー (Materialized View) の使用を制御します。「materialized_view_optimization オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **oem_string データベース・オプション**　oem_string データベース・オプションを使用して、データベース・ファイルのヘッダ・ページに情報を格納できます。この文字列は、アプリケーションからアクセスでき、バージョン情報を格納したり、そのデータベース・ファイルがアプリケーション用であることを検証したりするために使用できます。「oem_string オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **request_timeout データベース・オプション**　このオプションでは、接続が長時間にわたってサーバ・リソースを大量に消費しないように、単一要求を実行できる最大時間を指定します。「request_timeout オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **synchronize_mirror_on_commit オプション**　非同期モードまたは非同期フルページ・モードでデータベース・ミラーリングを実行しているときに、データベースの変更がミラー・サーバに送信されたことを確定するタイミングを制御します。「synchronize_mirror_on_commit オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **uuid_has_hyphens データベース・オプション**　uniqueidentifier 値が文字列に変換されるときのフォーマットを制御します。「uuid_has_hyphens オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **verify_password_function データベース・オプション**　verify_password_function データベース・オプションは、パスワード・ルールを実装するために使用できる関数を指定します。この関数は GRANT CONNECT 文で呼び出します。「verify_password_function オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **Java サポート用の新しいデータベース・オプション**　次のデータベース・オプションが追加されました。
  - 「java_location オプション [データベース]」『SQL Anywhere サーバ - データベース管理』
  - 「java_main_userid オプション [データベース]」『SQL Anywhere サーバ - データベース管理』
  - 「java_vm_options オプション [データベース]」『SQL Anywhere サーバ - データベース管理』

データベース・サーバ・オプション

「新機能」の項で説明されているサーバ・オプションのほかに、次の新しいサーバ・オプションが追加されています。

- **-cm サーバ・オプション**　このサーバ・オプションを使用して、Windows で Address Windowing Extensions (AWE) に割り当てるアドレス空間の大きさを指定できます。「-cm サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **-dh サーバ・オプション**　このサーバ・オプションを使用すると、サーバに対してサーバ列挙ユーティリティ (dblocate) を実行した場合でも、データベースが検出されないようになります。「-dh データベース・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **-dt サーバ・オプション**　このサーバ・オプションを使用して、テンポラリ・ファイルが格納されるディレクトリを指定できます。UNIX で共有メモリ接続を使用するデータベース・サーバでは、このオプションを指定できません。「-dt サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **-gtc サーバ・オプション**　このオプションを使用して、CPU で同時実行できるスレッド数を制御できます。「-gtc サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **-ot サーバ・オプション**　このサーバ・オプションを指定すると、メッセージが書き込まれる前にコンソール・ログ・ファイルがトランケートされます。「-ot サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **-su サーバ・オプション**　このオプションを使用すると、ユーティリティ・データベースへの接続時に DBA ユーザのパスワードを指定できます。*util_db.ini* ファイルの代わりに -su を使用する必要があります。「-su サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **-zp サーバ・オプション**　このサーバ・オプションを使用すると、最後に使用したクエリ最適化プランが接続ごとに格納されます。「-zp サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

プロパティとパフォーマンス・モニタの統計値

データベースを管理できるようにするための、接続、サーバ、データベースの新しいプロパティと、パフォーマンス・モニタの新しい統計値が追加されました。

- **接続プロパティ**　今回のリリースには、次の接続プロパティが追加されています。
  - allow_snapshot_isolation
  - ansi_substring
  - ApproximateCPUTime
  - conn_auditing
  - default_dbspace
  - ExprCacheAbandons
  - ExprCacheDropsToReadOnly
  - ExprCacheEvicts
  - ExprCacheHits
  - ExprCacheInserts
  - ExprCacheLookups
  - ExprCache
  - GetData
  - HeapsCarver
  - HeapsLocked
  - HeapsQuery
  - HeapsRelocatable
  - HttpServiceName
  - http_session_timeout
  - java_location
  - java_main_userid
  - LastPlanText
  - LockCount
  - LockedCursorPages
  - LockTableOID
  - materialized_view_optimization
  - max_query_tasks
  - max_temp_space
  - MultiPageAllocs
  - NcharCharSet
  - oem_string
  - post_login_procedure
  - QueryHeapPages
  - ReqCountActive
  - ReqCountBlockContention
  - ReqCountBlockLock
  - ReqCountBlockIO
  - ReqCountUnscheduled
  - ReqTimeActive
  - ReqTimeBlockContention
  - ReqTimeBlockIO

- ReqTimeBlockLock
- ReqTimeUnscheduled
- ReqStatus
- RequestsReceived
- RetryConnectionTimeout
- SessionCreateTime
- SessionID
- SessionLastTime
- SnapshotCount
- synchronize_mirror_on_commit
- tsql_outer_joins
- verify_password_function

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」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **サーバ・プロパティ**　今回のリリースには、次のサーバ・プロパティが追加されています。

  - CachePinned
  - CacheReadEng
  - CacheSizingStatistics
  - CarverHeapPages
  - ConsoleLogMaxSize
  - CollectStatistics
  - DebuggingInformation
  - DefaultNcharCollation
  - DiskReadEng
  - ExchangeTasks
  - FirstOption
  - FunctionMaxParms
  - FunctionMinParms
  - HeapsRelocatable
  - HeapsLocked
  - HeapsQuery
  - HeapsCarver
  - IsEccAvailable
  - IsRsaAvailable
  - LastConnectionProperty
  - LastDatabaseProperty
  - LastOption
  - LastServerProperty
  - MapPhysicalMemoryEng
  - MaxConnections
  - MultiProgrammingLevel
  - NumLogicalProcessors
  - NumLogicalProcessorsUsed
  - NumPhysicalProcessors
  - NumPhysicalProcessorsUsed

- QueryHeapPages
- RememberLastPlan
- RemoteputWait
- RequestFilterConn
- RequestFilterDB
- RequestLogMaxSize
- RequestsReceived
- ServerName
- StartDBPermission

これらのプロパティの詳細については、「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **データベースのプロパティ**　今回のリリースには、次のデータベース・プロパティが追加されています。

  - AccentSensitive
  - AlternateServerName
  - ArbiterState
  - AuditingTypes
  - CleanablePagesAdded
  - CleanablePagesCleaned
  - EncryptionScope
  - http_session_timeout
  - IOParallelism
  - JavaVM
  - LockCount
  - MirrorState
  - NcharCollation
  - NcharCharSet
  - NextScheduleTime
  - PartnerState
  - ReceivingTracingFrom
  - SendingTracingTo
  - SnapshotCount
  - SnapshotIsolationState
  - VersionStorePages
  - XPathCompiles

  これらのプロパティの詳細については、「データベース・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **パフォーマンス・モニタの統計値のプロパティ**　今回のリリースには、次のパフォーマンス・モニタの統計値が追加されています。

  - キャッシュ：マルチページ割り当て
  - キャッシュ：パニック
  - キャッシュ：スカベンジ・アクセス
  - キャッシュ：スカベンジ

- キャッシュ・ページ：割り当て構造体
- キャッシュ・ページ：ファイル
- キャッシュ・ページ：ファイル・ダーティ
- キャッシュ・ページ：空き
- 通信：受信要求数
- ヒープ：カーバ
- ヒープ：クエリ処理
- ヒープ：再配置可能ロック
- ヒープ：再配置可能
- メモリ・ページ：カーバ
- メモリ・ページ：固定カーソル
- メモリ・ページ：クエリ処理
- バージョン・ストア・ページ

これらの統計値の詳細については、「パフォーマンス・モニタの統計値」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

システム・プロシージャと関数

新しいシステム・プロシージャと関数、および既存のシステム・プロシージャと関数への新しい拡張機能を次に示します。

- **DEFAULT 句をサポートするためのすべてのプロシージャと関数の強化**　プロシージャとユーザ定義の関数では、値 DEFAULT に対応するパラメータがデフォルト値として定義されている場合、DEFAULT を引数として使用できます。プロシージャに複数のパラメータがあり、デフォルトに設定されていないパラメータがある場合は、名前付きのパラメータを使用するよりも、引数リストで DEFAULT を指定する方が簡単な場合があります。また、名前付きのパラメータは、関数呼び出しで使用できません。
- **新しいシステム・プロシージャ**　次のシステム・プロシージャが追加されました。
  - **sa_clean_database システム・プロシージャ**　データベース・クリーナが実行される期間を設定します。「sa_clean_database システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **sa_column_stats システム・プロシージャ**　sa_column_stats システム・プロシージャは、指定されたカラムについて文字列に関連する統計値を返します。「sa_column_stats システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **sa_conn_list システム・プロシージャ**　sa_conn_list システム・プロシージャは、接続 ID を返します。「sa_conn_list システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **sa_conn_options システム・プロシージャ**　sa_conn_options システム・プロシージャは、データベース・オプションに対応する接続プロパティのプロパティ情報を返します。「sa_conn_options システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **sa_db_list システム・プロシージャ**　sa_db_list システム・プロシージャは、データベース ID を返します。「sa_db_list システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **sa_describe_query システム・プロシージャ**　sa_describe_query システム・プロシージャは、カラムごとに 1 つのローを返し、結果の式とその null 入力属性のドメインを記述します。このプロシージャは、各カラムで EXPRTYPE 関数を実行することと同じです。「sa_describe_query システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **sa_get_bits システム・プロシージャ**　sa_get_bits システム・プロシージャは、ビット文字列を復号化し、ビットの値を示すビット文字列の各ビットについて 1 つのローを返します。「sa_get_bits システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **sa_make_object システム・プロシージャ**　sa_make_object システム・プロシージャで、イベントをオブジェクト・タイプとして指定できるようになりました。「sa_make_object システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **sa_materialized_view_info システム・プロシージャ**　sa_materialized_view_info システム・プロシージャは、指定した実体化ビュー (Materialized View) の情報 (ステータス、ビューの所有者など) を返します。「sa_materialized_view_info システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **sa_refresh_materialized_views システム・プロシージャ**　sa_refresh_materialized_views システム・プロシージャは、データベース内で現在初期化されていないすべての実体化ビュー (Materialized View) をリフレッシュします。「sa_refresh_materialized_views システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **sa_remove_tracing_data システム・プロシージャ**　このプロシージャは、診断トレーシング・テーブルから指定したロギング・セッションのすべてのレコードを永続的に削除します。「sa_remove_tracing_data システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **sa_save_trace_data システム・プロシージャ**　このプロシージャは、テンポラリ・トレーシング・テーブルからベース・テーブルにデータを保存します。「sa_save_trace_data システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **sa_set_tracing_level システム・プロシージャ**　プロファイルされるデータベースに対して生成されるトレーシング・データのレベルを設定します。「sa_set_tracing_level システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **sa_snapshots システム・プロシージャ**　データベースで現在アクティブなスナップショットのリストを返します。「sa_snapshots システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **sa_split_list システム・プロシージャ**　値のリストを表す文字列を引数に取り、そのリストを含む結果セットを返します。「sa_split_list システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **sa_table_stats システム・プロシージャ**　各テーブルから読み込まれたページ数に関する情報を返します。「sa_table_stats システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **sa_transactions**　データベースに対して現在実行しているトランザクションのリストを返します。「sa_transactions システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **sa_unload_cost_model と sa_load_cost_model システム・プロシージャ**　新しいシステム・プロシージャ sa_unload_cost_model と sa_load_cost_model を使用して、コスト・モデルをデータベースからアンロードし、別のデータベースにロードできるようになりました。これによって、同じようなハードウェア・インストールが大量にある場合に、繰り返し行われる時間のかかる再調整作業がなくなりました。「sa_unload_cost_model システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』と「sa_load_cost_model システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **新しい関数**　次の関数が追加されました。
  - **BIT_LENGTH 関数**　配列内に格納されたビット数を返します。「BIT_LENGTH 関数 [ビット配列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **BIT_SUBSTR 関数**　ビット配列のサブ配列を返します。「BIT_SUBSTR 関数 [ビット配列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **BIT_AND 関数**　2 つのビット配列を引数に取り、引数のビット処理 AND の結果を返します。比較する各ビットで両方のビットが 1 の場合は 1 を、それ以外の場合は 0 を返すという論理が使用されます。「BIT_AND 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **BIT_OR 関数**　2 つのビット配列を引数に取り、引数のビット処理 OR の結果を返します。比較する各ビットで片方 (または両方) のビットが 1 の場合は 1 を、それ以外の場合は 0 を返すという論理が使用されます。「BIT_OR 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **BIT_XOR 関数**　2 つのビット配列を引数に取り、引数のビット処理排他 OR の結果を返します。比較する各ビットで片方のビットだけ (両方のビットではなく) が 1 の場合は 1 を、それ以外の場合は 0 を返すという論理が使用されます。「BIT_XOR 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **COUNT_SET_BITS 関数**　配列で 1 (TRUE) に設定されたビットの数を返します。「COUNT_SET_BITS 関数 [ビット配列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **GET_BIT 関数**　ビット配列で指定したビットの値 (1 または 0) を返します。「GET_BIT 関数 [ビット配列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **REVERSE 関数**　この新しい関数は、文字式のリバースを返します。「REVERSE 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **SET_BIT 関数**　ビット配列の特定ビットの値を設定します。「SET_BIT 関数 [ビット配列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **SET_BITS 関数**　指定したビット (ロー・セットからの値に対応するビット) が 1 (TRUE) に設定されたビット配列を作成します。「SET_BITS 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **TRACED_PLAN 関数**　トレーシング・データと、クエリがトレースされたときのオプティマイザの状態に関する情報を使用して、クエリのグラフィカルなプランを生成します。「TRACED_PLAN 関数 [その他]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **さまざまなシステム・プロシージャと関数の強化**　次のシステム・プロシージャと関数が強化されました。
  - **プロパティ関数の強化**　プロパティ関数が LONG VARCHAR を返すようになりました。

    次の項を参照してください。

    - 「CONNECTION_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』
    - 「DB_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』
    - 「PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』
  - **DB_EXTENDED_PROPERTY 関数の強化**　DB_EXTENDED_PROPERTY 関数で NextScheduleTime データベース・プロパティを使用して、イベントの次にスケジュールされている実行時刻を取得できるようになりました。CHAR 文字セットの拡張情報を返す関数も使用できます。「DB_EXTENDED_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **新しい CONNECTION_EXTENDED_PROPERTY 関数**
    CONNECTION_EXTENDED_PROPERTY 関数を使用して、特定の接続パラメータの拡張情報を検索できます。「CONNECTION_EXTENDED_PROPERTY 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **sa_procedure_profile システム・プロシージャ**　sa_procedure_profile システム・プロシージャでは、出力をファイルに保存できるようになりました。また、新しい構文が追加され、いくつかのパラメータが必須になり、新しい使用方法が追加されました。「sa_procedure_profile システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **sa_procedure_profile_summary システム・プロシージャ**　sa_procedure_profile_summary システム・プロシージャでは、出力をファイルに保存できるようになりました。また、新しい構文が追加され、いくつかのパラメータが使用できるようになり、新しい使用方法が追加されました。「sa_procedure_profile_summary システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **sa_server_option システム・プロシージャ**　sa_server_option システム・プロシージャを使用すると、データベース・サーバの実行中に設定を変更できます。次の設定を変更できるようになりました。
    - **CacheSizingStatistics プロパティ**　キャッシュ・サイズが変更されるたびに、サーバ・メッセージ・ウィンドウにキャッシュ情報を表示します。

- **CollectStatistics プロパティ**　データベース・サーバのパフォーマンス・モニタの統計値を収集します。
- **ConsoleLogFile プロパティ**　サーバ・メッセージ・ウィンドウの情報が記録される出力ファイルの名前を指定します。
- **ConsoleLogMaxSize プロパティ**　サーバ・メッセージ・ウィンドウ情報の記録に使用される出力ファイルの最大サイズを指定します。
- **DebuggingInformation プロパティ**　トラブルシューティング目的で診断通信メッセージやその他のメッセージを表示します。
- **IdleTimeout サーバ・オプション**　指定された時間の間、要求を送信しなかった TCP/IP 接続または SPX 接続を切断します。
- **ProfileFilterConn プロパティ**　その他の接続によるデータベースの使用を妨げることなく、特定の接続 ID のプロファイリング情報を取得します。
- **RequestFilterDB プロパティ**　sa_server_option システム・プロシージャを使用して、要求ロギング用の単一データベースに対する接続をフィルタできるようになりました。
- **RequestLogging プロパティ**　要求ログでブロック・イベント、アンブロック・イベント、プラン情報、プロシージャ、トリガを記録できるようになりました。
- **RequestTiming プロパティ**　要求タイミングをオンにすると、各要求のタイミング情報を管理するように、データベース・サーバに指示されます。

詳細については、「sa_server_option システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **xp_startsmtp システム・プロシージャの強化**　xp_startsmtp システム・プロシージャでは、smtp_user_name、smtp_auth_username、smtp_auth_password の新しい 3 つのパラメータサポートされています。「xp_startsmtp システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **xp_sendmail システム・プロシージャの強化**　xp_sendmail システム・プロシージャが、include_file パラメータを使用することで、SMTP を使用してメールを送信するときに添付ファイルをサポートするようになりました。また xp_sendmail では、content_type パラメータを使用することで、SMTP メールを使用するときの MIME コンテンツをサポートします。「xp_sendmail システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **sa_conn_info システム・プロシージャが新しいプロパティ値を返す**　sa_conn_info システム・プロシージャが、追加プロパティ ClientPort、ServerPort、LockTable を返すようになりました。また、LastIdel プロパティを返さなくなり、UncmtOps 値の名前は UncommitOps に変更されました。「sa_conn_info システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **sa_performance_diagnostics がより多くの情報を返す**　スナップショット・アイソレーションの使用時に、sa_performance_diagnostics システム・プロシージャが LockCount と SnapshotCount を返すようになりました。「sa_performance_diagnostics システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **HASH 関数の強化**　HASH 関数で新しいアルゴリズム SHA256、SHA1_FIPS、SHA256_FIPS を使用できるようになりました。FIPS 関連のアルゴリズムは、FIPS 承認ソフトウェアを使用するシステムのみで使用されます。「HASH 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **COMPRESS 関数と DECOMPRESS 関数で新しいアルゴリズムをサポート**　関数で文字列を圧縮および解凍するために gzip アルゴリズムを使用できるようになりました。「COMPRESS 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』と「DECOMPRESS 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

SQL 文

次に、新しい SQL 文と、既存の SQL 文の構文に対する新しい拡張機能について説明します。これらの新機能は、このマニュアルの以前の機能の項に示されている文の変更に追加されています。

- **実体化ビュー (Materialized View) をサポートする SQL 文**　実体化ビュー (Materialized View) をサポートするために、次の SQL 文が追加または構文や機能が拡張されました。
  - 「ALTER MATERIALIZED VIEW 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「COMMENT 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「CREATE INDEX 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「CREATE MATERIALIZED VIEW 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「DROP 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「REFRESH MATERIALIZED VIEW 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「VALIDATE 文」『SQL Anywhere サーバ - SQL リファレンス』

  SELECT 文の新しい OPTION 句は、materialized_view_optimization データベース・オプションを上書きするために使用できます。「SELECT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **診断トレーシングとアプリケーション・プロファイリングをサポートする新しい SQL 文**　アプリケーション・プロファイリングをサポートするための新しい SQL 文は次のとおりです。
  - 「ATTACH TRACING 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「DETACH TRACING 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「REFRESH TRACING LEVEL 文」『SQL Anywhere サーバ - SQL リファレンス』
- **新しい VALIDATE DATABASE 文**　VALIDATE DATABASE 文を使用してデータベースを検証できるようになりました。「VALIDATE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **新しい VALIDATE MATERIALIZED VIEW 文**　VALIDATE MATERIALIZED VIEW 文を使用して、実体化ビュー (Materialized View) を検証できるようになりました。「VALIDATE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **新しい ALTER STATISTICS 文**　ALTER STATISTICS 文を使用して、カラムの統計値が自動的に更新されるようにするかを制御できるようになりました。自動更新が無効の場合でも、明

示的に CREATE STATISTICS 文または DROP STATISTICS 文を使用することで、統計値を強制的に更新できます。「ALTER STATISTICS 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **ALTER INDEX 文の強化** ALTER INDEX 文の REBUILD 句を使用して、インデックスを再構築できるようになりました。「ALTER INDEX 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **ALTER TABLE 文と CREATE TABLE 文の強化** MATCH 句を使用して、参照元テーブルの外部キーと参照先テーブルのプライマリ・キーの間で一致させる内容を詳細に制御できるようになりました。また、外部キーをユニークとして宣言できるようになり、一意性を個別に宣言する必要がなくなりました。「CREATE TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』と「ALTER TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **ALTER DATABASE 文の新しい CALIBRATE PARALLEL READ 句** 並列入出力が可能なハードウェアを検出するには、ALTER DATABASE 文の新しい CALIBRATE PARALLEL READ 句を使用します。dbspace の調整結果を取得するには、DB_EXTENDED_PROPERTY 関数を使用して新しい IOParallelism 拡張データベース・プロパティを問い合わせます。「ALTER DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』と「データベース・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **COMMENT 文の PRIMARY KEY ON 句** COMMENT 文の PRIMARY KEY ON 句を使用して、プライマリ・キーに注釈を作成できるようになりました。「COMMENT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **暗号化キーを変更するための CREATE ENCRYPTED FILE 文の強化** CREATE ENCRYPTED FILE 文の拡張機能を使用して、データベースのアンロードと再ロードを行わずにデータベース、トランザクション・ログ、dbspace の暗号化に使用する暗号化キーを変更できるようになりました。データベースが暗号化されていないがテーブルの暗号化が有効な場合は、CREATE ENCRYPTED FILE 文を使用してテーブルの暗号化に使用するキーを変更できます。「CREATE ENCRYPTED FILE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **CREATE DATABASE 文の強化** 文字セットの処理を向上させるために、新しい句 ENCODING、NCHAR COLLATION、ACCENT が追加されました。また、データベースの初期サイズを指定できるように、DATABASE SIZE 句が追加されました。「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **SELECT 文の強化** カーソルによるローの更新に使用される FOR UPDATE 句が拡張され、カラム・リストでは、後置された UPDATE 文を使用して変更可能なカラムを制限することができるようになりました。「SELECT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  SELECT 文の FROM 句が拡張され、READPAST テーブル・ヒントと UPDLOCK テーブル・ヒントがサポートされるようになりました。READPAST テーブル・ヒントは、データベース・サーバに対してロックされたローを無視するように指示します。UPDLOCK テーブル・ヒントは、XLOCK と同様に動作します。「FROM 句」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

SELECT 文が拡張され、その文のクエリ最適化を制御する OPTION 句がサポートされるようになりました。OPTION 句には、その SELECT 文の MATERIALIZED VIEW OPTIMIZATION 句による実体化ビュー (Materialized View) の一致を制御するための構文が含まれます。2 つ目の句 FORCE OPTIMIZATION は、コストベースの最適化をバイパスするようにクエリが修飾されている場合であっても、データベース・サーバに対してクエリの最適化を実行するように指示します。「SELECT 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **LOAD TABLE 文と UNLOAD TABLE 文の強化** LOAD TABLE 文の STRIP 句では、引用符で囲まれていない値を挿入する前に、その値から先行ブランクを削除するかを制御できるオプションを使用できるようになりました。追加の STRIP オプションを使用すると、データが削除される方法を微調整できます。

  LOAD TABLE 文が拡張され、COMMENTS INTRODUCED BY オプションがサポートされるようになりました。このオプションを使用して、入力データ内のコメントを識別するための文字列を指定できます。ロード操作時に、入力で指定した文字列で始まる行はすべて無視されます。

  LOAD TABLE 文と UNLOAD TABLE 文が拡張され、次のオプションがサポートされるようになりました。

  - **ENCODING オプション** データのロードやアンロードのときに使用するエンコーディングを指定するのに使用されます。
  - **ROW DELIMITED BY オプション** データのバルク・ロードやバルク・アンロードのときの入力レコードの末尾を示す文字列を指定するのに使用されます。
  - **QUOTE オプション** Interactive SQL の OUTPUT 文の QUOTE オプションに似ています。「OUTPUT 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

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

- **VALIDATE INDEX 文の強化** VALIDATE INDEX の構文が強化され、インデックス指定がサポートされるようになりました。「VALIDATE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **プライマリ・キーの名前を変更するための ALTER INDEX 文の強化** ALTER INDEX 文を使用して、プライマリ・キーの名前を変更できるようになりました。「ALTER INDEX 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **新しい CONTINUE 文** この文は、ループを再起動するために使用します。CONTINUE 文に続くループ内の文はスキップされます。「CONTINUE 文 [T-SQL]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **新しい BREAK 文 [T-SQL]** この文は、複合文またはループから出るために使用します。「BREAK 文 [T-SQL]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **INSERT 中にデフォルト値を更新する INSERT 文制御の強化** DEFAULTS ON | OFF 句を使用すると、ローがすでに存在するときに INSERT 中にデフォルト値が更新されるかを制御できます。DEFAULT TIMESTAMP、DEFAULT UTC TIMESTAMP、DEFAULT LAST USER の各デ

フォルト・フィールドには、この新機能は適用されません。これらのフィールドは常時更新されます。「INSERT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **ORDER BY 句をサポートするための DELETE 文の強化** DELETE 文で ORDER BY 句がサポートされ、データベースからローを削除する順序を指定できるようになりました。「DELETE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **START DATABASE 文の強化** START DATABASE 文は、データベースが起動できなかった理由を示せなかったときに、幅広いエラー・メッセージを返すようになりました。また、START DATABASE 句は任意の順序で指定できるようになりました。「START DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **イベント・ログやシステム・ログのみへのロギングをサポートするための MESSAGE 文の強化** ロギングのオン／オフを制御できるだけでなく、イベント・ログやシステム・ログのみにロギングするかを指定できます。MESSAGE 文の構文が拡張され、TO LOG 句内でオプション句 [EVENT | SYSTEM] を使用できるようになりました。たとえば TO EVENT LOG は、イベント・ログのみにロギングできます。「MESSAGE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **FOR OLAP WORKLOAD オプション** CREATE INDEX、CREATE TABLE、ALTER TABLE の各文の構文が拡張され、外部キー定義で FOR OLAP WORKLOAD オプションがサポートされるようになりました。このオプションは、OLAP パフォーマンスを向上するために、特定の最適化を実行してキーの統計値を収集するように、データベース・サーバに対して指示します。「CREATE INDEX 文」『SQL Anywhere サーバ - SQL リファレンス』、「CREATE TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』、「ALTER TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』、「クラスタード・ハッシュ GROUP BY アルゴリズム」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **テンポラリ・ストアド・プロシージャのサポート** CREATE PROCEDURE 文の拡張機能を使用して、テンポラリ・ストアド・プロシージャを作成できるようになりました。テンポラリ・ストアド・プロシージャは、そのストアド・プロシージャを作成した接続のみが参照でき、接続が切断されると自動的に切断されます。「CREATE PROCEDURE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **ローカル・テンポラリ・テーブルのサポート** CREATE LOCAL TEMPORARY TABLE 文を使用して、ローカル・テンポラリ・テーブルを作成できるようになりました。この方法で作成されたローカル・テンポラリ・テーブルは、接続が閉じると切断されます。「CREATE LOCAL TEMPORARY TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **テンポラリ・テーブルの強化** CREATE LOCAL TEMPORARY TABLE 文の SHARE BY ALL 句を使用して、データベースへのすべての接続でデータが共有されるグローバル・テンポラリ・テーブルを作成できるようになりました。「CREATE TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

データ型

- **CHAR と VARCHAR データ型の文字長セマンティックのサポート** CHAR または VARCHAR カラムを指定するときに、文字長セマンティックを使用できるようになりました。文字長セマ

ンティックを使用すると、長さをバイト数ではなく文字数で表現できます。「CHAR データ型」『SQL Anywhere サーバ - SQL リファレンス』と「VARCHAR データ型」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **ビット配列データ型のサポート** SQL Anywhere で VARBIT と LONG VARBIT データ型がサポートされるようになりました。これらのデータ型は、ビット配列を格納するために使用されます。「ビット配列データ型」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  ビット配列データ型で使用される次の関数が追加されました。

  - 「BIT_LENGTH 関数 [ビット配列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「BIT_SUBSTR 関数 [ビット配列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「BIT_AND 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「BIT_OR 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「BIT_XOR 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「COUNT_SET_BITS 関数 [ビット配列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「GET_BIT 関数 [ビット配列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「SET_BIT 関数 [ビット配列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「SET_BITS 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』

プログラミング・インタフェース

- **ADO.NET 2.0 のサポート** ADO.NET ドライバが更新され、.NET Framework のバージョン 2.0 がサポートされるようになりました。その一環で、新しいクラスとメソッドが追加されました。「SQL Anywhere .NET 2.0 API リファレンス」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **SQL Anywhere Explorer** SQL Anywhere Explorer を使用すると、Visual Studio .NET 内から SQL Anywhere データベースに接続できます。また、Sybase Central と Interactive SQL を Visual Studio .NET から直接開くことができます。「Visual Studio .NET でのデータベース接続の使用」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **PreparedStatement.addBatch メソッドのサポート** iAnywhere JDBC ドライバで PreparedStatement.addBatch メソッドがサポートされるようになりました。このメソッドはバッチ (またはワイド) 挿入を実行するときに便利です。

- **iAnywhere JDBC ドライバで JDBC 3.0 のサポート** iAnywhere JDBC ドライバで JDBC 3.0 呼び出しがサポートされるようになりました。「JDBC の概要」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **ODBC ドライバに SQL_GUID のサポート追加** SQL Anywhere ODBC ドライバに、UNIQUEIDENTIFIER カラムのサポートが追加されました。UNIQUEIDENTIFIER カラムは SQL_GUID として入力できるようになりました。

- **ODBC ドライバに GUID エスケープ・シーケンスのサポート追加** SQL Anywhere ODBC ドライバに、GUID エスケープ・シーケンスのサポートが追加されました。GUID エスケープ・シーケンスは、ODBC 経由で準備および実行された SQL 文で使用できます。GUID エスケープ・シーケンスの形式は、{guid 'nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn'} です。

- **ODBC メッセージ・コールバックが接続ごとになった**　ODBC では、Adaptive Server Anywhere バージョン 9.0.0 からメッセージ・コールバックをサポートしていますが、すべての接続のメッセージは単一のコールバック関数に送られていました。バージョン 9.0.2 からは、メッセージ・コールバック関数を指定すると、単一の接続のみに適用されるようになりました。これは、DBLIB の動作方法と一貫性があります。すべてのメッセージが、ODBC ドライバの単一関数に集まるようになりました。この関数は、メッセージを接続ごとにフィルタし、コールバック関数を備えた接続に対して、そのコールバック関数だけを呼び出します。

- **SQL Anywhere PHP モジュールに追加された新しい関数**　SQL Anywhere PHP モジュールに追加された新しい関数は次のとおりです。

  - sqlanywhere_execute
  - sqlanywhere_error
  - sqlanywhere_errorcode
  - sqlanywhere_insert_id

  また、2 つの新しいオプション verbose_errors と row_counts が sqlanywhere_set_option 関数に追加されました。「SQL Anywhere PHP API リファレンス」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **db_locate_servers_ex 関数の強化**　db_locate_servers_ex 関数で、2 つの新しいフラグ DB_LOOKUP_FLAG_ADDRESS_INCLUDES_PORT と DB_LOOKUP_FLAG_DATABASES がサポートされるようになりました。DB_LOOKUP_FLAG_ADDRESS_INCLUDES_PORT は、コールバック関数に渡された a_server_address 構造体内の TCP/IP ポート番号を返します。DB_LOOKUP_FLAG_DATABASES は、見つかったデータベースまたはデータベース・サーバごとにコールバック関数を 1 回呼び出すことを指定します。「db_locate_servers_ex 関数」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **Perl DBI モジュールの Perl DBD::ASAny ドライバ名の変更**　Perl ドライバの名前が DBD::ASAny から DBD::SQLAnywhere に変更されました。SQL Anywhere を使用する Perl スクリプトは、新しいドライバ名を使用するように変更する必要があります。ネイティブ SQL Anywhere 型を返すカーソル属性 ASATYPE は、変更されていません。また、型名がありません (ASA_STRING、ASA_FIXCHAR、ASA_LONGVARCHAR など)。「SQL Anywhere Perl DBD::SQLAnywhere API」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **SQL プリプロセッサ (sqlpp) の -o オプション値**　sqlpp の -o オプションで、Microsoft Windows 用に、WINNT ではなく WINDOWS を使用できるようになりました。また、サポートされている 64 ビットの UNIX オペレーティング・システム用に、UNIX64 を指定できるようになりました。「SQL プリプロセッサ」『SQL Anywhere サーバ - プログラミング』を参照してください。

新しい ODBC ドライバ・マネージャと ODBC ドライバの強化

- **ODBC ドライバ・マネージャの強化**　ODBC ドライバ・マネージャで、すべての ODBC 3.x 呼び出し、ワイド CHAR エントリ・ポイント、接続トレーシングがサポートされるようになりました。また、ODBC ドライバ・マネージャを使用して、非スレッド化またはスレッド化された SQL Anywhere ドライバ間で切り替えできるようになりました。

- **スレッド化アプリケーションと非スレッド化アプリケーションの両方での ODBC ドライバ・マネージャの使用**　スレッド化アプリケーションと非スレッド化アプリケーションの両方で、ODBC ドライバ・マネージャを使用できるようになりました。

配備

- **[配備] ウィザード**　[配備] ウィザードが追加され、SQL Anywhere for Windows の配備を作成できるようになりました。[配備] ウィザードを使用すると、Microsoft Windows Installer パッケージ・ファイルと Microsoft Windows Installer Merge Module ファイルの両方を作成できます。以前のバージョンの SQL Anywhere で提供されていた InstallShield のマージ・モジュールとテンプレートは、提供されなくなりました。代わりに、[配備] ウィザードを使用して、SQL Anywhere の配備を作成してください。「配備ウィザードの使用」『SQL Anywhere サーバ - プログラミング』を参照してください。

Windows CE の強化

- **WindowsCE での動的キャッシュ・サイズ決定のサポート**　WindowsCE で動的キャッシュ・サイズ決定がサポートされるようになりました。Windows や UNIX のように、Windows CE でも、データベース・サーバの負荷とその他のシステム・メモリへの要求に応じて、データベース・サーバのキャッシュ・サイズが増減します。この機能を使用すると、さまざまな状況で明示的にキャッシュ・サイズを選択する必要がなくなり、パフォーマンスが向上します。「動的キャッシュ・サイズ決定 (Windows)」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **Windows CE 用プロキシ・ポートの作成**　以前のリリースのソフトウェアでは、Windows CE デバイス上のデータベースに接続するためにプロキシ・ポートを使用するように ActiveSync を設定するには、レジストリのエントリを変更する必要がありました。Interactive SQL、Sybase Central、SQL Anywhere コンソール・ユーティリティの接続ダイアログに、Windows CE プロキシ・ポートの設定ツールが含まれるようになり、レジストリを変更しなくても、Windows CE デバイス上のデータベースに接続するためのプロキシ・ポートを作成できるようになりました。「Windows CE デバイス用プロキシ・ポートの作成」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **Windows CE 上のデータベースを再構築可能**　dbunload ユーティリティを使用してデータベースをファイルにアンロードし、そのファイルを新しいデータベースに再ロードするという 2 段階の処理で、Windows CE 上のデータベースを再構築できるようになりました。「Windows CE のデータベースの再構築」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **dbrunsql ユーティリティ**　SQL Anywhere スクリプト実行ユーティリティ (dbrunsql) では、Windows CE 上で SQL コマンドを入力したりコマンド・ファイルを実行したりすることができます。「SQL Anywhere スクリプト実行ユーティリティ (dbrunsql)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **Windows Mobile 5 用の署名済み .cab ファイル**　SQL Anywhere インストールには、VeriSign によって署名された、構築済みの *.cab* ファイルが含まれます。これらの *.cab* ファイルを使用した場合、発行元が不明であることを知らせる警告は表示されません。「インストール時の考

慮事項： Windows Mobile 5 への SQL Anywhere 10 の配備」 『SQL Anywhere サーバ - データベース管理』を参照してください。

UNIX／Linux の強化

- **UNIX プラットフォームで使用できる新しい ODBC ドライバ・マネージャ** UNIX プラットフォームで ODBC ドライバ・マネージャとして libdbodm10 共有オブジェクトを使用できるようになりました。iAnywhere ODBC ドライバ・マネージャを使用するアプリケーションでは、依存する ODBC をバージョン 3.0 以上に制限する必要があります。「UNIX 上での ODBC ドライバ・マネージャの使用」 『SQL Anywhere サーバ - プログラミング』を参照してください。
- **-uf サーバ・オプション** -uf オプションを使用すると、UNIX で致命的なエラーが発生したときにデータベース・サーバが対処する方法を指定できます。「-uf サーバ・オプション」 『SQL Anywhere サーバ - データベース管理』を参照してください。
- **Linux でサービス・ユーティリティのサポート** サービス・ユーティリティを Linux で使用して、SQL Anywhere サービスを作成、削除、リスト、開始、停止できるようになりました。
- **UNIX でサポートされる追加サンプル** UNIX でサポートされるようになった SQL Anywhere サンプルは次のとおりです。
  - **DBTools** このサンプルは、SQL Anywhere サンプル・データベースのバックアップを作成することで、データベース・ツール・ライブラリを呼び出してコンパイルする方法を説明するデータベース・ツール・アプリケーションです。
  - **DiskFull** Disk-Full Callback サンプル DLL を説明するためのサンプルです。
  - **HTTP** HTTP ディレクトリにあるサンプルでは、Web サービスのさまざまな機能の例が示されています。これには、Web サービスを使用して SQL プロシージャ内のクライアント側 cookie を設定および取得する方法、HTTP Web サービス・プロシージャ内のバイナリ・データを処理する方法、フォームや HTML テーブルを使用して簡単なカレンダーを表示する方法、xp_read_file を使用してローカル・ディスクからイメージを取得し、HTTP 要求への応答として返す方法、HTTP セッションを作成、使用、削除する方法などがあります。
  - **oemString** OEM ソフトウェア用にデータベース・ファイルが設定されているかどうかを判断する方法を説明するためのサンプルです。
  - **PerformanceFetch** fetchtest を使用して任意のクエリのフェッチ速度をテストする方法を説明するためのサンプルです。
  - **PerformanceInsert** instest を使用してテーブルへの挿入速度をテストする方法を説明するためのサンプルです。
- **Linux でファイバ (スレッド親和性) のサポートと同時実行処理の向上** Linux データベース・サーバ用の SQL Anywhere に、Windows ファイバの処理モデルに似た、新しいコルーチン処理モデルが導入されました。この処理モデルにより、データベース操作とのよりよい親和性を提供するタスクやルーチンを切り替えるような場合に、サーバをより詳細に制御できます。

- **Linux でのダイレクト I/O のサポート**　Linux 版 SQL Anywhere で、Linux 2.6 O_DIRECT 機能がサポートされるようになりました。この機能を使用すると、ファイル・システムが I/O をキャッシュしなくなるため、I/O パフォーマンスが向上します。
- **Linux での非同期 I/O**　Linux 版 SQL Anywhere で、Linux AIO 機能がサポートされるようになりました。この機能を使用すると、単一のアプリケーション・スレッドでも I/O 操作を他の処理と重複できるようになるため、I/O パフォーマンスが向上します。
- **Linux のデスクトップ GUI**　Linux 版 SQL Anywhere で、パーソナル・データベース・サーバ、ネットワーク・データベース・サーバ、Mobile Link サーバ、Mobile Link 同期クライアント、SQL Remote 用の追加デスクトップ GUI が提供されるようになりました。この GUI は、GTK ライブラリがインストールされている場合に、-ui オプションで呼び出すことができます。
- **Linux のデスクトップ・アイコン**　Linux 版 SQL Anywhere では、Linux デスクトップを使用するユーザが、データベース・サーバ、Sybase Central、Interactive SQL、SQL Anywhere コンソール・ユーティリティ、Mobile Link モニタの開始と管理をしやすくするために、Linux デスクトップ用にオプションでインストールできるアイコンが提供されるようになりました。
- **IBM AIX、HPUX、Sun Solaris に対する 64 ビット・クライアントのサポート**　64 ビット・メモリ・モデルで SQL Anywhere クライアント・ライブラリを使用できるようになり、IBM AIX、HP-UX、Sun Solaris のアプリケーションで大量のメモリを使用できるようになりました。

Web サービス

- **Web サーバの HTTP 1.1 への準拠**　HTTP 1.1 への準拠により、Web サーバで次の項目を受け入れるようになりました。
  - HTTP 要求のパイプライン処理。GET や HEAD などの複数の HTTP 要求を同時に実行できます。
  - 絶対 URI (以前は相対 URI のみサポートされていました)。
  - 100-continue request-header フィールド。クライアントは要求本文全体を送信する前に、サーバが要求を受け入れるかを (要求ヘッダを基に) 判断できます。
  - Accept-Charset request-header フィールドの quality 値 (以前はこれらの値は無視されていました)。
- **HTTP クライアントの HTTP 1.1 への準拠**　実装された HTTP 関連の強化は次のとおりです。
  - **HTTP 文字列メモリ・プール処理のサポート**　HTTP 文字列は連続したメモリに格納されなくなりました。キャッシュがバックエンド記憶領域として使用されます。
  - **クライアントのチャンク・モード**　HTTP クライアントは HTTP チャンク・モードを使用して POST 要求を送信できるようになりました。
  - **HTTP セッション**　HTTP 接続で、HTTP 要求間のステータスを管理する HTTP セッションを作成できるようになりました。
- **HTTP サーバの keep-alive オプション**　データベース・サーバで、HTTP クライアントからの要求時に keep-alive オプションがサポートされるようになりました。要求を終えるごとに接

続を閉じるのではなく、要求と応答を終えても HTTP 接続を開いたままにしておくことができるため、同じ接続で複数の要求を実行できます。「HTTP ヘッダの使用」『SQL Anywhere サーバ - プログラミング』を参照してください。

この機能をサポートするために、KeepaliveTimeout プロトコル・オプションも追加されました。「KeepaliveTimeout プロトコル・オプション [KTO]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しい HttpServiceName 接続プロパティ** 新しい接続プロパティ HttpServiceName が追加され、Web アプリケーションでサービス名のオリジンを判断できるようになりました。このプロパティは、エラー・レポートやフロー制御の場合に便利です。「接続レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **sa_set_http_option の強化** sa_set_http_option システム・プロシージャを使用して、要求の Accept-Charset request-header フィールドを基に、HTTP 応答で使用される文字セットを制御できるようになりました。「sa_set_http_option システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **SOAP サービスでのデータ型指定のサポート** CREATE SERVICE 文と ALTER SERVICE 文が拡張され、新しい DATATYPE 句がサポートされるようになりました。この句は、SOAP サービスでのみ使用するもので、入力パラメータと出力応答でデータ型指定をサポートするかどうかを制御します。「CREATE SERVICE 文」『SQL Anywhere サーバ - SQL リファレンス』と「ALTER SERVICE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **sa_set_soap_header システム・プロシージャ** sa_set_soap_header システム・プロシージャを使用して、SOAP サービス用の応答ヘッダを設定します。「sa_set_soap_header システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **SOAP_HEADER 関数と NEXT_SOAP_HEADER 関数** SOAP_HEADER 関数を使用して、SOAP サービスの要求ヘッダを取得します。「SOAP_HEADER 関数 [SOAP]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  NEXT_SOAP_HEADER 関数を使用して、SOAP ヘッダ内の次のヘッダ・エントリを取得します。「NEXT_SOAP_HEADER 関数 [SOAP]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **CREATE PROCEDURE 文、ALTER PROCEDURE 文、CREATE FUNCTION 文、ALTER FUNCTION 文の HEADER 句** これらの文には新しく HEADER 句が追加されています。この句は、HTTP Web サービスのクライアント・プロシージャと関数を作成するときに使用されます。この句を使用すると、HTTP 要求のヘッダ・エントリを追加または修正できます。

  「CREATE PROCEDURE 文」『SQL Anywhere サーバ - SQL リファレンス』、「CREATE FUNCTION 文」『SQL Anywhere サーバ - SQL リファレンス』、「HTTP ヘッダの使用」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **CREATE PROCEDURE 文、ALTER PROCEDURE 文、CREATE FUNCTION 文、ALTER FUNCTION 文の SOAPHEADER 句** これらの文には新しく SOAPHEADER 句が追加されています。この句は、SOAT Web サービスのクライアント・プロシージャと関数を作成するときに使用されます。この句では、IN (IN/OUT) 代入パラメータを使用して、送信する SOAP ヘッダ・エントリと受信する SOAP ヘッダ・データを指定できます。

「CREATE PROCEDURE 文」『SQL Anywhere サーバ - SQL リファレンス』、「CREATE FUNCTION 文」『SQL Anywhere サーバ - SQL リファレンス』、「SOAP ヘッダの使用」『SQL Anywhere サーバ - プログラミング』を参照してください。

その他

- **インデックス処理の強化**　今回のリリースでは、インデックス処理が次のように強化されています。
  - **インデックス共有処理のサポート**　プライマリ・キー、セカンダリ・キー、外部キー、または一意性制約を作成するときに、物理インデックス (ディスク上の実際のインデックス構造) を指す論理インデックスを作成できるようになりました。データベース・サーバは、論理インデックスを満たすために新しい物理インデックスが必要かを自動的に判断します。このモデルを使用すると、物理インデックスの共有処理が可能になり、重複した物理インデックスを作成したり管理したりすることがなくなり、ディスク領域を無駄にすることがなくなります。「論理インデックスを使用したインデックスの共有」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
  - **インデックス情報の記憶領域の向上**　データベースでインデックス情報が編成される方法が向上されました。たとえばプライマリ・キーと外部キーのインデックスを含むすべてのインデックスのリストが、単一のシステム・テーブル ISYSIDX に格納されます。

    3 つの新しいシステム・テーブル ISYSPHYSIDX、ISYSIDXCOL、ISYSFKEY によって、ISYSIDX でリストされるインデックスの追加情報が提供されます。次の項を参照してください。

    - 「カタログ内のインデックス情報」『SQL Anywhere サーバ - SQL の使用法』
    - 「SYSIDX システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
    - 「SYSPHYSIDX システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
    - 「SYSINDEXES システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
    - 「SYSFKEY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
  - **インデックス・コンサルタントの強化**　インデックス・コンサルタントが強化され、クラスタード・インデックス、負荷時のデータベースとサーバのステータス、負荷統計値の完全なレポートに関する推奨事項が改善されました。インデックス・コンサルタントはアプリケーション・プロファイリング・ツールに統合されました。
  - **インデックスの作成方法の制御が向上**　アプリケーションが参照整合性制約 (プライマリ・キー、外部キー、または一意性の制約) を作成すると、データベース・サーバは制約のキーを構成するカラムでインデックスを暗黙的に作成することで、その制約を確保します。データベース・サーバで、インデックスの作成方法を指定できるようになりました。制約キーのカラムの順序を指定したり、インデックスの各カラムに値のシーケンス (昇順または降順) を指定できます。また、外部キーのカラムの順序やシーケンスを、対応するプライマリ・キーや一意性制約と一致させる必要はありません。

    そのほかに、次の機能も強化されています。

    - プライマリ・キーの順序を変更するときに、テーブルのカラムを並べ替える必要がなくなりました。

- すべての制約インデックスのカラムのシーケンスを、アプリケーションの要件を満たすように指定できるようになりました。
- プライマリ・テーブルの設計と関連させなくても、外部キーのインデックスを外部キー・テーブルに関するアプリケーションの要件に適合させることができるようになりました。
- 外部キーに一意性制約を設定できるようになりました。

- **新しい外部ジョインの削除によるリライト最適化** 外部ジョインがクエリから削除されてもそのクエリが元のクエリと意味が変わらないときは、クエリの実行前に外部ジョインが削除されます。「セマンティック・クエリ変形」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **日付フォーマット文字列での文字長セマンティックの使用** 日付フォーマット文字列で、フォーマット指定子に置き換えるテキストの量を制御するために、文字長セマンティックを使用するようになりました。以前はバイト長セマンティックが使用されていました。たとえば文字列 MMM を使用して日付をフォーマットする場合、以前は月を格納するために 3 バイトを使用することを示していましたが、3 文字を意味するようになりました。
- **ディレクトリ・アクセス・サーバ** ディレクトリ・アクセス・サーバを作成することで、データベース・サーバを実行しているコンピュータのディレクトリ構造にアクセスするリモート・サーバを作成できるようになりました。「ディレクトリ・アクセス・サーバの使用」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **ノルウェー語の照合** 1252NOR が追加され、ノルウェー語がサポートされるようになりました。ノルウェー語の Windows システムで照合が指定されていない場合、データベース・サーバは、新規データベースのデフォルトの照合として 1252NOR を選択します。「サポートされている照合と代替照合」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **UTF8BIN の照合** UTF8BIN 照合が追加され、バイナリ・データのソート処理が向上しました。この新しい照合は、廃止される UTF8 照合に代わるものです。「サポートされている照合と代替照合」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **サーバ・メッセージ・ウィンドウの強化** サーバ・メッセージ・ウィンドウが、次のように強化されました。
  - **新しいウィンドウ・タイトル・バーの右クリック・メニュー** サポートされるすべての Windows プラットフォーム (Windows CE を除く) で、サーバ・メッセージ・ウィンドウのタイトル・バーを右クリックすることで、[バージョン情報] または [メッセージ領域のクリア] を選択できるようになりました。[バージョン情報] を選択すると、データベース・サーバの情報が表示されます。[メッセージ領域のクリア] を選択すると、サーバ・メッセージ・ウィンドウのすべてのメッセージが消去されます。このウィンドウのレプリカ (コンソールの出力ログ、Sybase Central の [サーバ・メッセージと実行された SQL] ウィンドウ枠、SQL Anywhere コンソール・ユーティリティ) は、消去操作の影響を受けません。
  - **データベース・サーバで使用される環境変数がサーバ・メッセージ・ウィンドウにロギングされる** -ze サーバ・オプションを使用すると、サーバ・メッセージ・ウィンドウにデータベース・サーバの環境変数のリストが表示されます。この機能は NetWare または WindowsCE では使用できません。「-ze オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **開始時のウィンドウ最小化の制御**　デフォルトでは、データベース・サーバの開始時に、サーバ・メッセージ・ウィンドウは最小化されています。データベース・サーバの開始時に、サーバ・メッセージ・ウィンドウを最小化しない場合は、-qn オプションを指定できます。「-qn サーバ・オプション」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **テーブルの前回更新日時の追跡機能**　データベース・サーバが、前回テーブルが更新された日時を追跡できるようになりました。SYSTAB システム・ビューの新しい last_modified_at カラムを使用します。「SYSTAB システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **JDBC ドライバでの SQL Server Native Client ODBC ドライバのサポート**　JDBC ドライバは、ODBC ドライバが Microsoft SQL Server Native Client ODBC ドライバであるかを確認し、デフォルトの結果セット・タイプやその他の属性を適切に設定します。

- **ミラーリング中に別のサーバに切り替わったときに SNMP トラップ**　SQL Anywhere SNMP Extension Agent がミラーリング中のサーバに接続しているときに切断されて、新しい接続が再確立されたにもかかわらず、接続先が*別の*サーバである場合は、SNMP エージェントがトラップを送信するようになりました。

  このトラップは、元のサーバがダウンしていて、ミラーとして動作していたサーバがプライマリになったことを示します。「トラップの使用」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **要求のロギングの変更**　要求のログはカンマ区切りのテキスト・フォーマットで格納されるようになり、元のサイズの約 3 分の 1 にまで小さくなりました。また、可能な場合は通常の時刻エントリではなく、等号 (=、ログで直前のエントリと同じ時刻という意味) または *+nnn* (*nnn* はログで直前のエントリから経過したミリ秒数) として時刻が記録されるようになりました。また、追加情報も記録されるようになりました。たとえばクエリでは、独立性レベル、フェッチされたローの数、カーソル・タイプが記録されます。INSERT、UPDATE、DELETE 文の場合は、影響を受けたローの数と起動されたトリガの数が記録されます。「要求ロギング」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

  sa_get_request_times システム・プロシージャでは新しい要求のログ・フォーマットだけをサポートします。ただし、tracetime Perl スクリプト *tracetime.pl* は新旧両方の要求のログ・フォーマットを処理します。tracetime スクリプトでは、新しいフォーマットのログの方が高速に処理できます。要求のログのサイズが大きいと著しく高速になります。

- **ODBC ドライバの強化**　SQL Anywhere は、Adaptive Server Enterprise と DB2 のデータベースに接続するときに、新しいドライバを使用してリモート・データ・アクセスを行います。「Mobile Link、QAnywhere、リモート・データ・アクセスで使用される ODBC の変更」 155 ページを参照してください。

- **SQLANYSAMP10 環境変数**　SQLANYSAMP10 環境変数は、*demo.db* や *custdb.db* サンプル・データベースなどの SQL Anywhere 10 のサンプルがあるディレクトリのロケーションを指定します。「SQLANYSAMP10 環境変数」 『SQL Anywhere サーバ - データベース管理』を参照してください。

バージョン・サポート

この項では、サードパーティ・コンポーネントでサポートされる／サポートされないバージョン番号の変更について説明します。

- **バージョン 8 以前のデータベースのサポート**　Sybase Central、Interactive SQL、または SQL Anywhere コンソール・ユーティリティを使用する場合、Adaptive Server Anywhere 8.0.0 以前のバージョンで作成したデータベースはサポートされなくなりました。古いソフトウェアで作成し、新しいソフトウェアでアップグレードしたデータベースも含まれます。このようなデータベースをロードしようとすると、データベースの開始時にエラーになります。Sybase Central からこのようなデータベースに接続して、新しいバージョン 10 データベースにアンロードすることができます。「SQL Anywhere のアップグレード」 350 ページを参照してください。
- **jConnect バージョン 5.5 と 6.0.5 のサポート**　SQL Anywhere では、データベース・サーバへの接続で jConnect バージョン 5.5 と 6.0.5 がサポートされるようになりました (バージョン 5.5 は 9.0.2 でもサポートされていました)。jConnect は、個別のダウンロードとして、http://www.sybase.com/products/informationmanagement/softwaredeveloperkit/jconnect で入手できます。サポートされている機能については、jConnect のマニュアルを参照してください。
- **Intel x86 アーキテクチャのプロセッサの要件**　Intel x86 アーキテクチャでは、SQL Anywhere は Pentium 以降のプロセッサだけをサポートし、80386 や 80486 などの古いプロセッサでは起動できません。

動作の変更

次に、バージョン 10.0.0 で導入された SQL Anywhere データベースに加えられた変更を、カテゴリごとに示します。

その他

- **Adaptive Server Anywhere の名前の変更**　バージョン 10.0.0 では、Adaptive Server Anywhere の名前が SQL Anywhere に変更されました。
- **アップグレードの変更**　バージョン 9.0.2 以前のデータベースをバージョン 10 にアップグレードするために、[データベース・アップグレード] ウィザード、アップグレード・ユーティリティ (dbupgrad)、ALTER DATABASE UPGRADE 文は使用できません。以前のバージョンのデータベースをバージョン 10 にアップグレードするには、アンロードと再ロードを実行し、データベースを再構築する必要があります。「SQL Anywhere のアップグレード」 350 ページを参照してください。
- **パスワードの変更**　新しく作成されたデータベースでは、データベースでの設定にかかわらず、すべてのパスワードは大文字と小文字が区別されます。新しいデータベースのデフォルトの DBA パスワードは、**sql** です。

  既存のデータベースを再構築する場合、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。

- パスワードを大文字と小文字を区別しないデータベースに最初に入力した場合、そのパスワードの大文字と小文字は区別されません。
- パスワードを大文字と小文字を区別するデータベースに最初に入力した場合、大文字のパスワードと、大文字と小文字が混在したパスワードでは、大文字と小文字が区別されます。ただし、パスワードをすべて小文字で入力した場合、パスワードの大文字と小文字は*区別されません*。
- 既存のパスワードと新しいパスワードの両方に加えられた変更は、大文字と小文字が区別されます。

データベース・サーバは、SHA256 を使用してパスワードをハッシュするようになりました。古いデータベースから再ロードされるパスワードでは、古い (独自の) ハッシュ処理アルゴリズムがサポートされ続けますが、新しいパスワードはすべて SHA256 を使用します。

パスワードは UTF-8 で格納されるようになりました。そのため、異なる文字セットを使用するデータベースに再ロードされても、データベースは動作し続けます。

以前のリリースでは、Embedded SQL から接続すると、DBA パーミッションでデータベースに接続したあとで、同じデータベースに対して任意のユーザでパスワードを指定しなくても 2 つ目の接続を正常に確立できました。これが、接続ごとにパスワードを指定しなければならなくなりました。

- **ブランク埋め込みの変更**　SQL Anywhere の以前のリリースでは、ブランクを埋め込まれたデータベースで文字列を比較した場合のセマンティックは、比較される 2 つの文字列に無限のブランクが埋め込まれているかのようになっていました。バージョン 10 では、これらのセマンティックが変更され、各文字列の後続ブランクを無視して比較を行うようになりました。

  等価 (=) 比較と不等価 (<>) 比較の場合は、セマンティックに変更はありません。2 つの方法 (ブランクの埋め込みと後続ブランクの無視) とも同じ結果になります。ただし、不等号比較の場合は異なります。たとえば、2 バイトの文字列 'a*' があるとします ('*' は、ブランクの値より小さい、データベースの照合順の文字を表します)。SQL Anywhere の以前のバージョンでは、比較述部 **'a*' < 'a'** は TRUE を返していました。バージョン 10 では、短い文字列には比較前にブランクが埋め込まれないので、この述部は FALSE となります。

  ブランクの埋め込みの詳細については、「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバ - データベース管理』と「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **プロパティの戻り値の大文字と小文字**　サーバ・プロパティ (PROPERTY 関数で返される) は以前のバージョンでは YES または NO を返していましたが、Yes または No を返すようになりました。データベース・プロパティ (DB_PROPERTY 関数で返される) と接続プロパティ (CONNECTION_PROPERTY 関数で返される) は、以前のバージョンでは ON または OFF を返していましたが、On または Off を返すようになりました。この変更は、大文字と小文字を区別するデータベースや、大文字と小文字を区別する文字列比較を使用するアプリケーションに影響する可能性があります。

- **サーバ・プロパティの戻り値の変更**　以前のリリースでは、サーバ・プロパティ ConnsDisabled と RememberLastStatement は値 ON と OFF を返していました。これらは値 Yes と No を返すようになりました。「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **sasrv.ini ファイルのデフォルト・ロケーションの変更**　*sasrv.ini* のデフォルトのロケーションは、Windows では *%ALLUSERSPROFILE%¥Application Data¥SQL Anywhere 10*、UNIX では *~/.sqlanywhere10* です。以前のリリースでは、UNIX でのファイル名は *.sasrv.ini* でした。このファイルは、すべてのプラットフォームで、*sasrv.ini* になりました。

- **長い名前のデータベース・サーバへの接続**　Windows と UNIX では、バージョン 9.0.2 以前のクライアントは、40 バイトより長い名前が設定されたバージョン 10.0.0 以降のデータベース・サーバに接続できません。

- **データベース・サーバで文字セットの変換は常に有効**　文字セットの変換を有効／無効にする -ct データベース・サーバ・オプションはサポートされなくなりました。データベース・サーバで文字セット変換は常に有効になりました。ただしデータベース・サーバで変換が不要であると判断された場合は、使用されません。文字セット変換を無効にするには、クライアントから CharSet=none と指定します。「CharSet 接続パラメータ [CS]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Windows CE でサポートされていない文字セット変換**　文字セット変換は Windows CE ではサポートされていません。以前のリリースでは、文字セット変換は Windows CE 用のデータベース・サーバ上では無効になっており、データベースにはどの文字セットも使用することができました。今回のリリースでは、オペレーティング・システムの文字セットまたは UTF-8 のいずれかを使用して、Windows CE 用にデータベースを作成する必要があります。「インストール時の考慮事項： Windows CE での ICU の使用」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **システム・プロシージャと関数の変更**　システム・プロシージャと関数の変更は次のとおりです。

  - **複数のシステム・プロシージャの内部への組み込み**　外部システム・プロシージャの xp_read_file、xp_write_file、xp_sprintf、xp_scanf、xp_cmdshell が、内部システム・プロシージャになりました。

  - **sa_validate システム・プロシージャ**　sa_validate システム・プロシージャで、VALIDATE 権限が必要になりました。「sa_validate システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  - **sa_reset_identity システム・プロシージャ**　table-name パラメータが必須になりました。また、owner-name パラメータを指定しない場合は、table-name パラメータがデータベース内のテーブルをユニークに識別する必要があります。「sa_reset_identity システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  - **sa_locks システム・プロシージャ**　sa_lock システム・プロシージャの出力が変更され、追加情報 (接続 ID、ユーザ ID、テーブル名、ロック・クラス、ロック期間) を返すようになりました。「sa_locks システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  - **RAND 関数**　以前のバージョンでは、各接続のシードが同じ値に設定されていたため、RAND 関数が各接続で同一のシーケンスを返すことがありました。今回のリリースでは、各接続がさまざまなランダム・シーケンスを認識するため、各接続はユニークなシードが設定されるようになりました。「RAND 関数 [数値]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **コールバック関数 DB_CALLBACK_START と DB_CALLBACK_FINISH**　コールバック関数 DB_CALLBACK_START と DB_CALLBACK_FINISH がすべてのプラットフォームでサポートされるようになりました (以前は Windows プラットフォームのみでサポートされていました)。「db_register_a_callback 関数」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **UNC 名を使用して指定されたファイルで分散読み込みが使用されなくなった**　リモート・コンピュータ上のファイルや UNC 名 (たとえば *¥¥mycomputer¥myshare¥mydb.db*) で指定されたファイルに対して、分散読み込みが使用されなくなりました。「適切なページ・サイズの使用」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **プライマリ・キー制約と外部キー制約でのカラムの順序**　プライマリ・キー制約を作成するときは、任意の順序でカラムを指定できます。テーブルにカラムが出現する順序は関係ありません。また、外部キーのカラムとプライマリ・キーのカラムのマッピングを指定すれば、参照先のプライマリ・キーとカラムの順序が異なる外部キーを作成できるようになりました。「CREATE TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』の PRIMARY KEY 句を参照してください。

- **インデックスで重複したカラム名を使用できなくなった**　以前のバージョンでは、インデックスのカラムを重複参照できていましたが、プライマリ・キー、外部キー、一意性制約指定は除外されていました。今回のバージョンでは、すべてのタイプのインデックスで動作が一貫されるようになりました。カラム名を重複して指定すると、エラーが返されます。また、以前のデータベースに重複したカラム参照のあるインデックスが含まれると、dbunload ユーティリティによって reload.sql の生成時に重複したカラムがインデックスから削除されます。「CREATE TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **暗号化データベース・プロパティ**　SELECT DB_PROPERTY('Encryption') を実行すると、データベースが暗号化されていない場合でも、None 以外の値を返すようになりました。これは、データベースでテーブル暗号化が有効な場合に発生します。アプリケーションでデータベースが暗号化されているかを確認するメソッドとしてこのコマンドを実行する場合は、代わりに SELECT DB_PROPERTY('EncryptionScope') を使用してください。「データベース・プロパティの概要」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **FIPS を使用して HTTPS を開始する場合の構文の変更**　この場合、以前のバージョンでは -xs HTTPS_FIPS(...) を指定していました。今回のバージョンでは、-xs HTTPS(FIPS=yes;...) を指定する必要があります。以前の構文はサポートはされますが、廃止される予定です。「-xs サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **ユーザ ID の最大長は 128 バイト**　以前のリリースでは、文でユーザ ID が必要な場合、データベース・サーバでは 128 バイトを超えるユーザ ID を使用する前にトランケートしていました。string_rtruncation オプションを設定した場合は、トランケーション・エラーが返されていました。これが 128 バイトを超えるユーザ ID を指定した場合は、string_rtruncation オプションの設定に関係なく、データベース・サーバがエラーを返すようになりました。「識別子」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **サーバ名の最大長**　TCP/IP 接続と共有メモリ接続で、データベース・サーバ名の最大長が 40 バイトから 250 バイトに増加しました。「-n サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **識別子で使用可能な文字の変更**　識別子で二重引用符と円記号を使用できなくなりました。「識別子」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **LicensesInUse サーバ・プロパティ名の変更**　サーバ・プロパティ LicensesInUse の名前は、UniqueClientAddresses に変更されました。「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **SQL Anywhere OLE DB プロバイダ名の変更**　SQL Anywhere OLE DB プロバイダの名前は、以前は ASAProv、ASAProv.90、ASAProv.80 でしたが、SAOLEDB になりました。バージョン 10 のプロバイダは、特に SAOLEDB.10 という名前で参照できます。
- **SQL Anywhere サンプル・データベース ODBC DSN の変更**　ODBC データ・ソースの名前は、以前は ASA 9.0 Sample でしたが、SQL Anywhere 10 Demo になりました。
- **接続文字列の変更**　ODBC 接続と OLE DB 接続で、接続パラメータを検出する場所の優先度が接続文字列、SQLCONNECT 環境変数、データ・ソースの順になりました。以前のバージョンでは、ODBC 接続と OLE DB 接続でデータ・ソースの優先度は SQLCONNECT よりも高くなっていました。「接続パラメータについての注意」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **空の値を使用した接続パラメータが未指定として扱われるようになった**　すべての API で、空の値で指定された接続パラメータは、パラメータが指定されなかったものとして扱われます。以前のリリースでは空の値は、指定された場所と使用されている API に応じて未指定または空の文字列として扱われていました。「接続パラメータについての注意」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **監査がオンの場合はトランザクション・ログをオフにできない**　以前のバージョンのソフトウェアでは、監査がオンになっているデータベースでトランザクション・ログの使用を停止できました。今後は、データベースで監査がオンになると、トランザクション・ログを使用する必要があります。トランザクション・ログの使用を中止する場合は、監査をオフにする必要があります。
- **監査がオンになっているデータベースは読み込み専用モードで起動できない**　以前のバージョンのソフトウェアでは、監査がオンになっているデータベースを読み込み専用モードで開始できました。今後は、監査がオンになっているデータベースは、読み込み専用モードで起動できません。
- **符号付き BIGINT カラムの精度が 20 から 19 になった**　以前は、ODBC アプリケーションで SQL_BIGINT を使用して*符号付き* BIGINT カラムを記述すると、精度 20 の値が返されていました。今回のバージョンでは、精度 19 の値が返されるようになりました。以前の (不適切な) 値に依存するアプリケーションは変更する必要があります。
- **Java VM の強化**　SQL Anywhere では、Java オプションは別途ライセンスが必要なコンポーネントとして提供されなくなりました。データベース内の Java は、Java コードを実行するために内部 VM を使用するのではなく、外部 VM を使用するようになりました。これによって、任意の Java VM を使用できるようになり、特定の JDK バージョンや Java ターゲットに限定されなくなりました。新しく初期化されるデータベースは、常に Java 対応になります。

  その結果、次の変更が行われました。

- **未サポートのデータベース・オプション**　SQL Anywhere による以下のオプションのサポートは終了しました。
  - describe_java_format
  - java_heap_size
  - java_namespace_size
  - java_page_buffer_size
  - java_input_output
  - return_java_as_string
- **未サポートのプロパティ**　以下のプロパティは、サポートされなくなりました。
  - データベース・プロパティ
    - JDKVersion
    - JavaHeapSize
    - JavaNSSize
  - データベース・サーバ・プロパティ
    - IsJavaAvailable
    - JavaGlobFix
  - 接続プロパティ
    - JavaHeapSize
    - java_input_output
- **新しい JavaVM プロパティ**　JavaVM データベース・プロパティは、データベース内の Java を実行するためにデータベース・サーバが使用する Java VM のパスを返します。
- **未サポートの互換性ビュー・カラム**　システム互換性ビューで、次のカラムを使用できなくなりました。
  - SYSINFO.classes_version
  - SYSJAVACLASS.replaced_by
  - SYSJAVACLASS.type_id
- **データベース・ユーティリティで廃止された Java オプション**　次のデータベース・ユーティリティ・プロパティが廃止されました。
  - 初期化ユーティリティ (dbinit)：-ja、-jdk
  - アンロード・ユーティリティ (dbunload)：-jr
  - アップグレード・ユーティリティ (dbupgrad)：-ja、-jdk、-jr、-j
- **CREATE DATABASE 文と ALTER DATABASE 文の一部の Java 関連句で Java サポートが廃止された**　CREATE DATABASE 文では、JAVA ON | OFF 句と、JDK バージョン句がサポートされなくなりました。ALTER DATABASE 文では、REMOVE JAVA 句がサポートされなくなりました。
- **新しい Java ファイル**　前述の変更のほかに、*java¥sajvm.jar* ファイルが追加されました。

- **Ping ユーティリティ (dbping)** 以前は、データベース・サーバがプロパティ値に対して NULL を返すと、Ping ユーティリティ (dbping) はエラーを返していました。今回のバージョンでは、プロパティ値が不明の場合に、dbping は NULL を出力し、成功のリターン・コードで終了するようになりました。プロパティ値が不明の場合に dbping が失敗のエラー・コードで終了するようにする場合は、-en オプションを指定できます。「Ping ユーティリティ (dbping)」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **環境変数の名前の変更** 今回のリリースには、次の環境変数の名前が変更されています。

以前の名前	新しい名前
ASTMP	SATMP
ASDIR	SADIR
ASLOGDIR	SALOGDIR
ASLANG	SALANG
ASCHARSET	SACHARSET

- **PHP モジュールのファイル名の変更** PHP ファイルの命名規則が変更されました。以前のバージョンでは、ファイル名は *phpX_sqlanywhereY.dll* という形式で、*X* は PHP メジャー・バージョン番号、*Y* は SQL Anywhere のメジャー バージョン番号でした。今回のバージョンでは、*php-a.b.c_sqlanywhereY.dll* という形式で、*a.b.c* はファイルがビルドされた PHP ソースの完全なバージョン番号、*Y* は SQL Anywhere のメジャー・バージョン番号です。たとえば *php-5.0.2_sqlanywhere10.dll* のようになります。

- **PrefetchBuffer 接続パラメータの値の指定** PrefetchBuffer 接続パラメータが、下位互換のために 16384 以下の値をキロバイトとして解釈するようになりました。k サフィックスなしでキロバイトを使用する方法は廃止されます。PrefetchBuffer の値が有効な範囲外にあったり k サフィックスなしでキロバイト単位を指定したりしたためにこの値が調整される場合は、実際に使用された PrefetchBuffer 値がクライアント・ログ・ファイルに示されます。「PrefetchBuffer 接続パラメータ [PBUF]」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **システム定義のドメインを削除できない** システム定義のドメイン (MONEY、UNIQUEIDENTIFIERSTR など) はデータベースから削除できなくなりました。「DROP 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **データベース・ユーティリティの変更** 前述したとおり、データベース・ユーティリティの変更は次のとおりです。

  - **サービス・ユーティリティ (dbsvc) でサービスとしてログインする権限を付与できる** サービス・ユーティリティ (dbsvc) で -a オプションを使用し、サービスとしてログインする権限が有効でないアカウントで実行しようとすると、サービスとしてログインする権限を付与するかどうかを指定するプロンプトが表示されます。-y オプションを使用すると、dbsvc はプロンプトせずに、サービスとしてログインする権限を付与しようとします。「Windows 用サービス・ユーティリティ (dbsvc)」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **アンロード・ユーティリティ (dbunload) の -an オプションをリモート・サーバに使用できる**　この変更以前には、dbunload -an は同じコンピュータ上のサーバのみに対して実行できていました。今回のリリースでは、dbunload -an を別のコンピュータで実行しているサーバに対して実行できるようになりました。「アンロード・ユーティリティ (dbunload)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **サーバ列挙ユーティリティ (dblocate) ホスト名または IP アドレスのフォーマット**　ホスト名や IP アドレスは、-n を指定したかに関係なく、任意のフォーマットで使用できます。たとえばサーバが myhost.mycompany.com で実行しており、この IP アドレスが 1.2.3.4 の場合に、このコンピュータで実行しているサーバだけを mycompany.com ドメインの任意のコンピュータからリストするには、dblocate myhost、dblocate myhost.mycompany.com、dblocate 1.2.3.4 のいずれも使用できます。以前のバージョンでは、指定されたホスト名や IP アドレスは dblocate で表示されるアドレス文字列 (ポート番号を除く) に一致する必要があったため、dblocate myhost.mycompany.com または dblocate -n 1.2.3.4 だけが動作していました。「サーバ列挙ユーティリティ (dblocate)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **デフォルト関連の変更**　デフォルトは、次のように変更されています。
  - **パーソナル・データベース・サーバのデフォルト TCP/IP 受信アドレスの変更**　Windows では、パーソナル・データベース・サーバは 0.0.0.0 ではなく、127.0.0.1 で接続を受信するようになりました。この変更により、Windows ファイアウォールを有効にして SQL Anywhere を実行している場合は、dbeng10 の使用前に例外リストに追加する必要がなくなりました。

    これにより、*hostname* がコンピュータの実際のホスト名や IP アドレスの場合、LINKS=tcpip (HOST=*hostname*;DOBROADCAST=none) と接続しようとしても動作しません。ただし、ホスト名 **localhost** または **127.0.0.1** を使用すると動作します。
  - **デフォルト・データベース・ページ・サイズが 4096 に変更**　SQL Anywhere データベースのデフォルト・データベース・ページ・サイズが、4096 バイトから 2048 バイトに変更されました。このページ・サイズにすると、多くの環境でパフォーマンスが向上することがわかっています。「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』.

    -gp オプションを指定しないで、データベースがロードされていない状態でデータベース・サーバを起動した場合、そのデータベース・サーバのデフォルト・ページ・サイズは 4096 になります。
  - **デフォルト最大キャッシュ・サイズの変更**　Windows (非 AWE) のデフォルトの最大キャッシュ・サイズが増加しました。デフォルトの最大キャッシュ・サイズは次のどちらか小さい方になりました。
    - (*総物理メモリ量* - 4 MB) の 90%、2 MB 以上
    - (*使用可能なアドレス空間* - 512 MB)
  - **UNIX キャッシュ・サイズ**　UNIX での最大キャッシュ・サイズの計算方法が変更されました。デフォルトの最大キャッシュ・サイズは次のように計算されます。
    - 32 ビット UNIX プラットフォームでは、総物理メモリ量の 90% と 1,834,880 KB のどちらか小さい方

64 ビット UNIX プラットフォームでは、総物理メモリ量の 90% と 8,589,672,320 KB のどちらか小さい方

「キャッシュが使用するメモリの制限」『SQL Anywhere サーバ - SQL の使用法』と「-ch サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **UNIX ストアド・プロシージャ** 既存の UNIX アプリケーションをアップグレードするときに、64 ビット・データベース・サーバを使用している場合は、既存の外部ストアド・プロシージャを 64 ビットに変更する必要があります。
- **NULL 定数を NUMERIC または文字列データ型に変換する場合のデフォルト・サイズ** NULL 定数を NUMERIC データ型または文字列データ型 (CHAR、VARCHAR など) に変換する場合、長さが 32767 ではなく 0 に設定されるようになりました。
- **openxml システム・プロシージャのデフォルト URI の変更** openxml システム・プロシージャを使用する場合、名前空間宣言が指定されなかったときは、デフォルトでプレフィクス mp が Uniform Resource Identifier (URI) にバインドされます。以前のリリースでは、この URI は urn:ianywhere-com:asa-xpath-metaprop でした。この値が urn:ianywhere-com:sa-xpath-metaprop に変更されました。「openxml システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **-c、-ch-、-cl サーバ・オプションのキャッシュ・サイズの割合を計算する方法の変更** -c、-ch、または -cl を指定して P (パーセント) を使用すると、物理システム・メモリ量または使用可能なアドレス空間量のどちらか小さい方に対して割合が計算されるようになりました。アドレス割り当てに使用できるよりも多くのメモリをキャッシュに割り当てようとするリスクがなくなります。「-c サーバ・オプション」『SQL Anywhere サーバ - データベース管理』、「-ch サーバ・オプション」『SQL Anywhere サーバ - データベース管理』、「-cl サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **Procedure_profiling サーバ・オプション名の変更** プロシージャ・プロファイリングを制御するサーバ・オプションの正しい名前が ProcedureProfiling になりました。以前の Procedure_profiling も使用できますが、将来のリリースでサポートされなくなる予定です。「sa_server_option システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **デフォルト・ポートを使用していない HP-UX 上のデータベース・サーバに接続しているクライアントで、TCP/IP ポート番号の指定不要** 以前のリリースで、HP-UX でデータベース・サーバを起動する場合、デフォルト・ポート (2638) が使用中またはデフォルト・ポートを使用しないのであれば、ServerPort [PORT] プロトコル・オプションを使用してポート番号を指定する必要がありました。

  HP-UX では、1 つのマシンで複数のデータベース・サーバが起動している場合に、TCP/IP の ServerPort プロトコル・オプションは不要になりました。Mac OS X では、サーバが同じコンピュータですでに実行している場合にネットワーク・サーバを起動するには、依然として TCP/IP の ServerPort オプションを指定する必要があります。「ServerPort プロトコル・オプション [PORT]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **SOAP CONCRETE 応答の名前が ASADataSet から SimpleDataset に変更** CONCRETE 応答の名前が ASADataSet から SimpleDataset に変更されました。「CREATE SERVICE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **[データベース・アンロード] ウィザードの動作の変更**　データベースをバージョン 10 以前のデータベース・バージョンにアンロードできなくなりました。バージョン 9.0.2 以前のデータベースをバージョン 10 データベースにアンロードすると、再構築が完了してもデータベースに自動的に接続できません。
- **[データベースの抽出] ウィザードの動作の変更**　バージョン 9.0.2 以前のデータベースを抽出できません。バージョン 10 データベースから抽出する必要があります。
- **Solaris で -ui と -ux サーバ・オプションの未サポート**　Solaris で -ui と -ux サーバ・オプションはサポートされなくなりました。Linux では引き続き使用できます。
- **数値データ型の変換**　DOUBLE 型から NUMERIC に変換する場合に、元の DOUBLE 値の近似精度が最も高いアルゴリズムが使用されるようになりました。これらの変更により、有効桁数が 15 桁以下の DOUBLE 値は、正確に NUMERIC に変換されます。場合によっては、SQL Anywhere の以前のバージョンとは異なる結果になることがあります。「数値セット間の変換」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **sa_validate システム・プロシージャの変更**　sa_validate システム・プロシージャのデータ、インデックス、フルの各オプションは必要なくなり、使用できなくなる予定です。式やチェックサムの検証を要求しないかぎり、以前のデータ、インデックス、フルのオプションを使用して実行されていた検査は、デフォルトで実行されます。「sa_validate システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **a_validate_type 列挙の変更**　a_validate_type 列挙の VALIDATE_DATA、VALIDATE_INDEX、VALIDATE_FULL の各パラメータは必要なくなり、使用できなくなる予定です。VALIDATE_NORMAL が指定されると、これらのオプションで実行された検証は、デフォルトで実行されます。「a_validate_type 列挙」『SQL Anywhere サーバ - プログラミング』を参照してください。
- **SQLPATH 環境変数の構文の変更**　UNIX での SQLPATH 環境変数の構文が変更されました。以前のバージョンでは、すべてのオペレーティング・システムで、各パス要素をセミコロン (;) で区切っていました。SQL Anywhere 10 では、UNIX プラットフォームではパス要素をコロン (:) で区切り、その他のプラットフォームではセミコロンで区切ります。

データベース・オプションの変更

- **大文字と小文字の区別とデータベース・オプション**　SET OPTION 文と CONNECTION_PROPERTY 関数では、使用するオプション名の大文字と小文字を区別しません。ただし、トルコ語の照合を使用するデータベースや大文字と小文字を区別するデータベースでは、「アルファベット順のオプション・リスト」『SQL Anywhere サーバ - データベース管理』で指定された大文字と小文字を使用して、クエリで参照されるオプション名を記述する必要があります。

  このような場合、オプション名の大文字と小文字が正しくないと、SYSOPTION のクエリや次のようなクエリでは、どのローとも一致しない可能性があります。

```
SELECT *
  FROM sa_conn_properties()
  WHERE propname = 'BLOCKING'
```

- **ansi_blanks が On に設定された状態で Embedded SQL の使用**　ansi_blanks が On でブランクが埋め込まれたデータベースを使用する Embedded SQL では、データ型 DT_STRING の値を指定する場合は、sqllen フィールドを値が格納されたバッファの長さ (少なくとも値の長さと末尾の null 文字用のスペースの合計) に設定する必要があります。

  ブランクが埋め込まれたデータベースでは、ansi_blanks オプションによって、フェッチされた式が CHAR または NCHAR (VARCHAR または NVARCHAR ではない) で、その式が CHAR または NCHAR (VARCHAR または NVARCHAR ではない) ホスト変数にフェッチされている場合、クライアントに送信されるトランケーション警告が制御されます。「ansi_blanks オプション [互換性]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **ansi_integer_overflow オプションのデフォルト設定の変更**　新しいデータベースが作成されるときの、ansi_integer_overflow データベース・オプションのデフォルト値は On です。ソフトウェアの以前のバージョンでは、このオプションのデフォルト値は Off でした。「ansi_integer_overflow オプション [互換性]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **date_format オプションの変更**　date_format では、フォーマット文字列を指定するときに次の値をサポートしなくなりました。

  - **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**　午前／午後の識別子 (A.M. または P.M.、12 時間表記)
  - **aaa[a...]**　午前／午後の識別子 (A.M. または P.M.、12 時間表記)
  - **pp**　(必要に応じて) 午後の識別子 (P.M.、12 時間表記)
  - **ppp[p...]**　(必要に応じて) 午後の識別子 (P.M.、12 時間表記)

  また、文字データがマルチバイトの場合、各記号の長さが文字数を反映するようになりました。たとえば 'mmm' 記号は、月に 3 文字の長さを指定します。以前のバージョンでは、この記号の長さはバイト数を反映していました。「date_format オプション [互換性]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **login_mode データベース・オプション**　login_mode データベース・オプションの値 Mixed は廃止されます。標準ログインと統合化ログインの両方を可能にする場合は、Standard,Integrated を指定します。「login_mode オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **string_rtruncation オプションのデフォルト設定の変更**　新しいデータベースを作成するときの string_rtruncation データベース・オプションのデフォルト値は On です。ソフトウェアの以前のバージョンでは、このオプションのデフォルト値は Off でした。「string_rtruncation オプション [互換性]」『SQL Anywhere サーバ - データベース管理』を参照してください。

  CAST 関数を使用して文字列をトランケートする場合、string_rtruncation データベース・オプションは Off に設定する必要があります。それ以外の場合はエラーになります。

文字列のトランケートには LEFT 関数を使用することをおすすめします。「LEFT 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **temp_space_limit_check オプションのデフォルト設定の変更** temp_space_limit_check オプションのデフォルトの設定は On に変更されました。デフォルトでは、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 が返されるようになりました。「temp_space_limit_check オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **timestamp_format オプションの変更** timestamp_format オプションでは、フランス語の月日の使用がサポートされなくなりました。また、文字データがマルチバイトの場合、各記号の長さが文字数を反映するようになりました。たとえば 'mmm' 記号は、月に 3 文字の長さを指定します。以前のバージョンでは、この記号の長さはバイト数を反映していました。「timestamp_format オプション [互換性]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **truncate_date_values オプションの削除** truncate_date_values オプションは削除されました。以前のリリースでは、このオプションを使用すると DATE データ型を使用して定義されたカラムに時刻を含めることができました。今回のリリースでは、DATE で定義されたカラムには日付だけを含めることができます。日付と時刻を格納する場合は、TIMESTAMP データ型を使用してください。「TIMESTAMP データ型」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

サーバ・オプションの変更

- **-ec サーバ・オプションと -xs サーバ・オプションで強力な暗号化タイプの TLS 構文が変更** -ec サーバ・オプション、-xs サーバ・オプション、暗号化接続パラメータで、強力な暗号化タイプの構文が変更されました。none、simple、tls の 3 種類だけになりました。タイプとして使用するキー交換アルゴリズムを指定するのではなく、暗号化タイプとして tls を指定し、アルゴリズムの指定に新しいプロトコルオプション tls_type を使用できるようになりました。「-ec サーバ・オプション」『SQL Anywhere サーバ - データベース管理』、「-xs サーバ・オプション」『SQL Anywhere サーバ - データベース管理』、「Encryption 接続パラメータ [ENC]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **-os サーバ・オプション** 以前のリリースでは、-os データベース・サーバ・オプションにより、ログ・ファイルの名前が *current-file-name.old* に変更されました。これが -on オプションの動作となりました。-os データベース・サーバ・オプションでは、出力ログの最大サイズ (このサイズに達するとログの名前が変更される) を指定するようになりました。以前は、-os を使用すると 2 つのログ・ファイルが作成されていましたが、今回のバージョンでログ・ファイルの数が無制限になりました。「-os サーバ・オプション」『SQL Anywhere サーバ - データベース管理』と「-on サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

カタログの変更

カタログは、バージョン 10.0.0 で大きく変更されました。最も重要な変更は、システム・テーブルの名前が変更され、名前の先頭に I が含まれるようになったことです。システム・テーブルに

アクセスしようとすると、パーミッション拒否エラーを受け取ります。システム・テーブル内の情報は、システム・ビューから使用できます。システム・テーブルあたりシステム・ビューは 1 つあります。下位互換のためにこのシステム・ビューの名前は以前のバージョンの SQL Anywhere からのテーブル名と一致します。たとえば 9.0.2 では、SYS.SYSARTICLE と呼ばれるシステム・テーブルがありました。バージョン 10.0.0 では、このシステム・テーブルは SYS.ISYSARTICLE と呼ばれ、対応するシステム・ビューは SYS.SYSARTICLE になります。

カタログには、統合ビューも含まれるようになりました。これらのビューは、一般に必要な 2 つ以上のテーブルやビューからのジョインを提供します。統合ビューのほとんどは、以前のリリースでシステム・ビューとして存在していました。

一部のシステム・テーブルやシステム・ビューは、廃止、またはカタログから削除されました。ただしほとんどの場合、互換ビューは提供されていません。

Adaptive Server Anywhere 9.0.2 から SQL Anywhere 10.0.0 へのカタログの完全なマッピングを次の表に示します。1 番目のカラム「**9.0.2 システム・テーブル/ビュー**」には、9.0.2 システム・テーブルの名前と、スラッシュ (/) を挟んで 9.0.2 関連ビューを示します。中央のカラム「**10.0.0 システム・テーブル**」には、10.0.0 テーブル名が含まれます。最後のカラム「**10.0.0 システム・ビュー**」には、関連する 10.0.0 ビュー名と、互換性に関する注意が含まれます。

注意
カラム内のダッシュ (-) は、等価オブジェクトがないことを示します。たとえば 10.0.0 リリースのカタログで新しいテーブルは、9.0.2 カラムのテーブルの位置はダッシュになります。

9.0.2 システム・テーブル/ビュー	10.0.0 システム・テーブル	10.0.0 システム・ビュー
DUMMY / -	DUMMY	-
RowGenerator / -	RowGenerator	-
SYSARTICLE / SYSARTICLES	ISYSARTICLE	「SYSARTICLE システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』 10.0.0 以前と互換性を持たせる場合：「SYSARTICLES 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSARTICLECOL / SYSARTICLECOL	ISYSARTICLECOL	「SYSARTICLECOL システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』 10.0.0 以前と互換性を持たせる場合：「SYSARTICLECOLS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSATTRIBUTE / -	ISYSATTRIBUTE	-
SYSATTRIBUTENAME / -	ISYSATTRIBUTENAME	-

9.0.2 システム・テーブル/ビュー	10.0.0 システム・テーブル	10.0.0 システム・ビュー
SYSCAPABILITY / SYSCAPABILITIES	ISYSCAPABILITY	「SYSCAPABILITY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』 「SYSCAPABILITIES 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSCAPABILITYNAME / -	ISYSCAPABILITYNAME	「SYSCAPABILITYNAME システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
- / SYSCATALOG		「SYSCATALOG 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSCHECK / -	ISYSCHECK	「SYSCHECK システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
- / SYSCOLAUTH	-	「SYSCOLAUTH 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSCOLLATION / -	-	「SYSCOLLATION 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』
SYSCOLLATIONMAPPINGS / -	-	「SYSCOLLATIONMAPPINGS 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』
SYSCOLPERM / -	ISYSCOLPERM	「SYSCOLPERM システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSCOLSTAT / SYSCOLSTATS	ISYSCOLSTAT	「SYSCOLSTAT システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSCOLSTATS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSCOLUMN / SYSCOLUMNS	ISYSTABCOL	「SYSTABCOL システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSCOLUMNS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』 10.0.0 以前と互換性を持たせる場合：「SYSCOLUMN 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』
SYSCONSTRAINT / -	ISYSCONSTRAINT	「SYSCONSTRAINT システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
- / -	ISYSDEPENDENCY	「SYSDEPENDENCY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSDOMAIN / -	ISYSDOMAIN	「SYSDOMAIN システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』

9.0.2 システム・テーブル/ビュー	10.0.0 システム・テーブル	10.0.0 システム・ビュー
SYSEVENT / -	ISYSEVENT	「SYSEVENT システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
SYSEVENTTYPE / -	ISYSEVENTTYPE	「SYSEVENTTYPE システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
SYSEXTENT / -	-	-
SYSEXTERNLOGINS / -	ISYSEXTERNLOGIN	「SYSEXTERNLOGIN システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
SYSFILE / -	ISYSFILE	「SYSFILE システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
SYSFKCOL / -	ISYSIDXCOL	「SYSINDEXES システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』 10.0.0 以前と互換性を持たせる場合： 「SYSFKCOL 互換ビュー (旧式)」 『SQL Anywhere サーバ - SQL リファレンス』
SYSFOREIGNKEY / SYSFOREIGNKEYS	ISYSFKEY	「SYSFKEY システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』と「SYSFOREIGNKEYS 統合ビュー」 『SQL Anywhere サーバ - SQL リファレンス』 10.0.0 以前と互換性を持たせる場合： 「SYSFOREIGNKEY 互換ビュー (旧式)」 『SQL Anywhere サーバ - SQL リファレンス』
- / SYSGROUPS	ISYSGROUP	「SYSGROUP システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』と「SYSGROUPS 統合ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
SYSHISTORY / -	ISYSHISTORY	「SYSHISTORY システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
SYSINDEX / SYSINDEXES	ISYSIDX	「SYSIDX システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』と「SYSINDEXES 統合ビュー」 『SQL Anywhere サーバ - SQL リファレンス』 10.0.0 以前と互換性を持たせる場合： 「SYSINDEX 互換ビュー (旧式)」 『SQL Anywhere サーバ - SQL リファレンス』
SYSINFO / -	-	「SYSINFO 互換ビュー (旧式)」 『SQL Anywhere サーバ - SQL リファレンス』

9.0.2 システム・テーブル/ビュー	10.0.0 システム・テーブル	10.0.0 システム・ビュー
SYSIXCOL / -	ISYSIDXCOL	「SYSINDEXES システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』 10.0.0 以前と互換性を持たせる場合：「SYSIXCOL 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』
SYSJAR / -	ISYSJAR	「SYSJAR システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSJARCOMPONENT / -	ISYSJARCOMPONENT	「SYSJARCOMPONENT システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSJAVACLASS / -	ISYSJAVACLASS	「SYSJAVACLASS システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSLOGIN / -	ISYSLOGINMAP	「SYSLOGINMAP システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSOPTBLOCK / -	-	システムでのみ使用
- / -	ISYSMVOPTION	「SYSMVOPTION システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
- / -	ISYSMVOPTIONNAME	「SYSMVOPTIONNAME システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
- / -	ISYSOBJECT	「SYSOBJECT システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSOPTION / SYSOPTIONS	ISYSOPTION	「SYSOPTION システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSOPTIONS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSOPTJOINSTRATEGY / SYSOPTJOINSTRATEGIES	-	システムでのみ使用
SYSOPTORDER / SYSOPTORDERS	-	システムでのみ使用
SYSOPTQUANTIFIER / -	-	システムでのみ使用
SYSOPTREQUEST / -	-	システムでのみ使用
SYSOPTREWRITE / -	-	システムでのみ使用

9.0.2 システム・テーブル/ビュー	10.0.0 システム・テーブル	10.0.0 システム・ビュー
SYSOPTSTAT / -	ISYSOPTSTAT	「SYSOPTSTAT システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
-	ISYSPHYSIDX	「SYSPHYSIDX システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
- / SYSPROCAUTH	-	「SYSPROCAUTH 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSPROCEDURE / SYSPROCEDURES	ISYSPROCEDURE	「SYSPROCEDURE システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』 SYSPROCEDURES ビューは、SYSPROCS に名前が変更されました。「SYSPROCS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSPROCPARM / SYSPROCPARMS	ISYSPROCPARM	「SYSPROCPARM システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSPROCPARMS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSPROCPERM / -	ISYSPROCPERM	「SYSPROCPERM システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
-	ISYSPROXYTAB	「SYSPROXYTAB システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSPUBLICATION / SYSPUBLICATIONS	ISYSPUBLICATION	「SYSPUBLICATION システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSPUBLICATIONS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』
- / -	ISYSREMARK	「SYSREMARK システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSREMOTEOPTION / SYSREMOTEOPTIONS , SYSREMOTEOPTION2	ISYSREMOTEOPTION	「SYSREMOTEOPTION システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』、「SYSREMOTEOPTION2 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』、「SYSREMOTEOPTIONS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSREMOTEOPTIONTYPE / -	ISYSREMOTEOPTIONTYPE	「SYSREMOTEOPTIONTYPE システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』

9.0.2 システム・テーブル/ビュー	10.0.0 システム・テーブル	10.0.0 システム・ビュー
SYSREMOTETYPE / SYSREMOTETYPES	ISYSREMOTETYPE	「SYSREMOTETYPE システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』 と 「SYSREMOTETYPES 統合ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
SYSREMOTEUSER / SYSREMOTEUSERS	ISYSREMOTEUSER	「SYSREMOTEUSER システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』 と 「SYSREMOTEUSERS 統合ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
SYSSCHEDULE / -	ISYSSCHEDULE	「SYSSCHEDULE システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSSERVERS / -	ISYSSERVER	「SYSSERVER システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
- / -	ISYSSOURCE	「SYSSOURCE システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
SYSSQLSERVERTYPE / -	ISYSSQLSERVERTYPE	「SYSSQLSERVERTYPE システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
SYSSUBSCRIPTION / SYSSUBSCRIPTIONS	ISYSSUBSCRIPTION	「SYSSUBSCRIPTION システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』 と 「SYSSUBSCRIPTIONS 統合ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
SYSSYNC / SYSSYNCS, SYSSYNC2	ISYSSYNC	「SYSSYNC システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』、「SYSSYNC2 統合ビュー」 『SQL Anywhere サーバ - SQL リファレンス』、「SYSSYNC2 統合ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
-	ISYSSYNCSCRIPT	「SYSSYNCSCRIPT システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』 と 「SYSSYNCSCRIPTS 統合ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
- / SYSSYNCSUBSCRIPTIONS	-	「SYSSYNCSUBSCRIPTIONS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』
- / SYSSYNCUSERS	-	「SYSSYNCUSERS 統合ビュー」 『SQL Anywhere サーバ - SQL リファレンス』

9.0.2 システム・テーブル/ビュー	10.0.0 システム・テーブル	10.0.0 システム・ビュー
- / SYSTABAUTH	-	「SYSTABAUTH 統合ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
SYSTABLE / -	ISYSTAB	「SYSTAB システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』 10.0.0 以前と互換性を持たせる場合：「SYSTABLE 互換ビュー (旧式)」 『SQL Anywhere サーバ - SQL リファレンス』
-	ISYSTABCOL	「SYSTABCOL システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
SYSTABLEPERM / -	ISYSTABLEPERM	「SYSTABLEPERM システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
SYSTRIGGER / SYSTRIGGERS	ISYSTRIGGER	「SYSTRIGGER システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』 と 「SYSTRIGGERS 統合ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
SYSTYPEMAP / -	ISYSTYPEMAP	「SYSTYPEMAP システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
-	ISYSUSER	「SYSUSER システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
- / SYSUSERAUTH	ISYSUSERAUTHORITY	「SYSUSERAUTHORITY システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』 と 「SYSUSERAUTH 互換ビュー (旧式)」 『SQL Anywhere サーバ - SQL リファレンス』
- / SYSUSERLIST		「SYSUSERAUTHORITY システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』 と 「SYSUSERLIST 互換ビュー (旧式)」 『SQL Anywhere サーバ - SQL リファレンス』
SYSUSERMESSAGES / -	ISYSUSERMESSAGE	「SYSUSERMESSAGE システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
- / SYSUSEROPTIONS	-	「SYSUSEROPTIONS 統合ビュー」 『SQL Anywhere サーバ - SQL リファレンス』

9.0.2 システム・テーブル/ビュー	10.0.0 システム・テーブル	10.0.0 システム・ビュー
SYSUSERPERM / SYSUSERPERMS	-	データは ISYSUSER と ISYSUSERAUTHORITY システム・テーブルに置かれるようになりました。「SYSUSER システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSUSERAUTHORITY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。 10.0.0 以前と互換性を持たせる場合：「SYSUSERPERM 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』と「SYSUSERPERMS 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』
SYSUSERTYPE / -	ISYSUSERTYPE	「SYSUSERTYPE システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
- / SYSVIEWS	ISYSVIEW	「SYSVIEW システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSVIEWS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』
SYSWEBSERVICE / -	ISYSWEBSERVICE	「SYSWEBSERVICE システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』

新しいビューの一覧

システム・ビュー名	詳細な情報
SYSDEPENDENCY	SYSDEPENDENCY システム・ビューの各ローは、2 つのデータベース・オブジェクト間の依存性を示します。「SYSDEPENDENCY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSFKEY	SYSFKEY の各ローは、システムの外部キー制約を示します。「SYSFKEY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSIDX	SYSIDX システム・テーブルの各ローは、データベースの論理インデックスを定義します。「SYSIDX システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSIDXCOL	SYSIDXCOL システム・ビューの各ローは、SYSIDX システム・ビューで記述されたインデックスのカラム 1 つを示します。「SYSINDEXES システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

システム・ビュー名	詳細な情報
SYSLOGINMAP	SYSLOGINMAP システム・ビューには、統合化ログインか Kerberos ログインを使用したデータベースへの接続に使用できるすべてのユーザ名が含まれます。「SYSLOGINMAP システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSMVOPTION	SYSMVOPTION システム・ビューの各ローは、実体化ビュー (Materialized View) のオプション値 1 つの設定を示します。「SYSMVOPTION システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSMVOPTIONNAME	SYSMVOPTIONNAME システム・ビューの各ローには、SYSMVOPTION システム・ビューで定義されたオプションの名前が含まれます。「SYSMVOPTIONNAME システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSOBJECT	SYSOBJECT システム・ビューの各ローは、オブジェクトを示します。データベース・オブジェクトには、テーブル、ビュー、カラム、インデックス、プロシージャなどがあります。「SYSOBJECT システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSPHYSIDX	SYSPHYSIDX システム・ビューの各ローは、データベースの物理インデックスを定義します。「SYSPHYSIDX システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSPROCS	SYSPROCS システム・ビューは、以前の SYSPROCEDURES ビューを置き換えます。「SYSPROCS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSPROXYTAB	SYSPROXYTAB システム・ビューの各ローは、プロキシ・テーブル 1 つのリモート・パラメータを示します。「SYSPROXYTAB システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSREMARK	SYSREMARK システム・ビューの各ローは、オブジェクトの注釈 (またはコメント) を示します。「SYSREMARK システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSSOURCE	SYSSOURCE システム・ビューの各ローには、ISYSOBJECT システム・テーブルにリストされたオブジェクトのソースが含まれます。「SYSSOURCE システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSSYNCSCRIPT	SYSSYNCSCRIPT システム・ビューの各ローは、Mobile Link スクリプト・アップロードのストアド・プロシージャを指定します。「SYSSYNCSCRIPT システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSTABCOL	SYSTABCOL システム・ビューには、データベース内のテーブルとビューそれぞれの各カラムのロー 1 つが含まれます。「SYSTABCOL システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

システム・ビュー名	詳細な情報
SYSUSER	SYSUSER システム・ビューの各ローは、データベース内のユーザを示します。「SYSUSER システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSUSERAUTHORITY	SYSUSERAUTHORITY システム・ビューの各ローは、ユーザ ID に付与された権限を示します。「SYSUSERAUTHORITY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

廃止されるテーブルやビューの一覧

廃止されるカタログ・オブジェクトを次に示します。ほとんどの場合、オブジェクトは以前のバージョンのテーブルでしたが、互換ビューになりました。これらのオブジェクトを参照してもエラーにはなりませんが、今後の互換性のために、推奨されるオブジェクトを指すようにアプリケーションを変更することをおすすめします。

廃止されるテーブルやビュー	変換情報
SYSCOLLATION システム・テーブル	照合マッピング情報はデータベース・プロパティとして格納されるようになりました。「SYSCOLLATION 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSCOLLATIONMAPPINGS システム・テーブル	照合マッピング情報はデータベース・プロパティとして格納されるようになりました。「SYSCOLLATIONMAPPINGS 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSCOLUMN システム・テーブル	代わりに SYSTABCOL システム・ビューを使用してください。「SYSTABCOL システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSCOLUMN 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSFKCOL システム・テーブル	代わりに SYSFKEY システム・ビューを使用してください。「SYSFKEY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSFKCOL 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSFOREIGNKEY システム・テーブル	代わりに SYSFKEY システム・ビューを使用してください。「SYSFKEY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSFOREIGNKEY 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSINDEX システム・テーブル	代わりに SYSIDX システム・ビューを使用してください。「SYSIDX システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSINDEX 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

廃止されるテーブルやビュー	変換情報
SYSIXCOL システム・テーブル	代わりに SYSIDXCOL システム・ビューを使用してください。「SYSINDEXES システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSIXCOL 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSTABLE システム・テーブル	代わりに SYSTAB システム・ビューを使用してください。「SYSTAB システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSTABLE 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSUSERAUTH システム・ビュー	代わりに SYSUSERAUTHORITY システム・ビューを使用してください。「SYSUSERAUTHORITY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSUSERAUTH 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSUSERPERM システム・テーブル	代わりに SYSUSERAUTHORITY システム・ビューを使用してください。「SYSUSERAUTHORITY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSUSERPERM 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSUSERLIST システム・ビュー	代わりに SYSUSERAUTHORITY システム・ビューを使用してください。「SYSUSERAUTHORITY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSUSERLIST 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSUSERPERMS システム・ビュー	代わりに SYSUSERAUTHORITY システム・ビューを使用してください。「SYSUSERAUTHORITY システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSUSERPERMS 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

削除または名前が変更されたテーブルやビューの一覧

カタログに存在しなくなったカタログ・オブジェクトを次に示します。これらのオブジェクトを参照すると、エラーになります。

削除されたテーブルやビュー	変換情報
SYSATTRIBUTE システム・テーブル	代わりに SYSTAB と SYSPHYSIDX システム・ビューを使用してください。空いている割合とクラスタード・インデックスの情報は、ISYSTAB システム・テーブルで保管されるようになりました。キーの値、キーの距離、リーフ・ページ、深さの情報は、ISYSPHYSIDX システム・テーブルで保管されるようになりました。「SYSTAB システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSPHYSIDX システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

削除されたテーブルやビュー	変換情報
SYSATTRIBUTENAME システム・テーブル	代わりに SYSIDX と SYSPHYSIDX システム・ビューを使用してください。「SYSIDX システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSPHYSIDX システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSEXTENT システム・テーブル	SQL Anywhere バージョン 10.0.0 以上のカタログでは、SYSEXTENT テーブルを使用できなくなりました。このテーブルは、以前のバージョンでは未使用でした。
SYSEXTERNLOGINS	SYSEXTERNLOGIN に名前が変更されました。「SYSEXTERNLOGIN システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSLOGIN システム・テーブル	SYSLOGIN テーブルは、一部変更され、SYSLOGINMAP システム・ビューに置き換えられました。「SYSLOGINMAP システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSOPTBLOCK	このテーブルは内部でのみ使用されていました。
SYSOPTJOINSTRATEGY	このテーブルは内部でのみ使用されていました。
SYSOPTJOINSTRATEGIES	このビューは内部でのみ使用されていました。
SYSOPTORDER	このテーブルは内部でのみ使用されていました。
SYSOPTORDERS	このビューは内部でのみ使用されていました。
SYSOPTQUANTIFIER	このテーブルは内部でのみ使用されていました。
SYSOPTREQUEST	このテーブルは内部でのみ使用されていました。
SYSOPTREWRITE	このテーブルは内部でのみ使用されていました。
SYSPROCEDURES ビュー	代わりに SYSPROCS 統合化ビューを使用してください。「SYSPROCS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSSERVERS	SYSSERVER に名前が変更されました。「SYSSERVER システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
SYSUSERMESSAGES	SYSUSERMESSAGE に名前が変更されました。「SYSUSERMESSAGE システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

システム・テーブルとシステム・ビューのカラムの変更

システム・テーブルとシステム・ビューのカラムにはさまざまな変更が加えられました。後述する変更以外は、新しいカラムの追加か未使用のカラムの削除で、どちらもアプリケーションへの影響はありません。

- **SYSCOLUMN と SYSCOLUMNS ビュー**　これらの両方のビューの width カラムが、SMALLINT から UNSIGNED INT に変更されました。「SYSCOLUMN 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』と「SYSCOLUMNS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **SYSCONSTRAINT ビュー**　以前の SYSCONSTRAINT システム・テーブルは、新しいシステム・テーブル ISYSCONSTRAINT と、対応する SYSCONSTRAINT システム・ビューで置き換えられました。SYSCONSTRAINT の参照で、新しいシステム・ビューが使用されるようになりました。これが今回のリリースで大きく異なる点です。SYSCONSTRAINT システム・ビューの内容を確認するには、「SYSCONSTRAINT システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **SYSREMOTEOPTION ビュー**　SYSREMOTEOPTION から選択できなくなりました。代わりに SYSREMOTEOPTIONS または SYSREMOTEOPTION2 を使用してください。「SYSREMOTEOPTIONS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』または「SYSREMOTEOPTION2 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **SYSJAR、SYSJARCOMPONENT、SYSJAVACLASS ビュー**　create_time カラムは削除されました。ただし作成時刻の情報は、SYSOBJECT.create_time で使用できます。「SYSOBJECT システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **SYSFILE システム・ビュー**　store_type カラムは INTEGER になりました。「SYSFILE システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **SYSPROCPARM と SYSLOGINMAP ビュー**　これらのビューから remarks カラムが削除されました。また、SYSPROCPARM の width カラムが、SMALLINT から UNSIGNED INT に変更されました。「SYSPROCPARM システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSLOGINMAP システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **SYSPROCPARMS ビュー**　SYSPROCPARM.width が、SMALLINT から UNSIGNED INT に変更されました。「SYSPROCPARMS 統合ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **SYSREMOTEUSER ビュー**　log_send、log_sent、confirm_sent、log_received、confirm_received の各カラムは UNSIGNED BIGINT になりました。「SYSREMOTEUSER システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **SYSSUBSCRIPTION ビュー**　created と started カラムは UNSIGNED BIGINT になりました。「SYSSUBSCRIPTION システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **SYSSYNC ビュー**　progress、created、log_sent カラムは UNSIGNED BIGINT になりました。「SYSSYNC システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

SQL 文

- **REVOKE CONNECT 文**　ユーザを削除するために REVOKE CONNECT 文を実行すると、指定したユーザが所有するオブジェクトがすべてユーザとともに削除されます。データベースに、削除対象のユーザが所有するオブジェクトに依存する、別のユーザが所有するアクティブなビューが含まれる場合、REVOKE CONNECT 文はエラーを返します。「REVOKE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **抽出テーブルのキー・ジョインの制限**　TOP N、START AT、FIRST、ORDER BY、Window 関数、FOR XML、または再帰テーブルを含む抽出テーブルでキー・ジョインを行えなくなりました。「ビューと抽出テーブルのキー・ジョイン」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **ALTER SERVER 文と CREATE SERVER 文**　サーバ・クラス ASAJDBC と ASAODBC は、それぞれ SAJDBC と SAODBC に名前が変更されました。「ALTER SERVER 文」『SQL Anywhere サーバ - SQL リファレンス』と「CREATE SERVER 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **ALTER 文**　すべての ALTER 文で、サブ句として MODIFY ではなく、ALTER を使用するようになりました。アプリケーションで MODIFY サブ句を使用している場合は、代わりに ALTER サブ句を使用するように変更する必要があります。MODIFY 構文は、サポートはされますが、廃止される予定です。この影響は次の文に及びます。
  - 「ALTER DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「ALTER EVENT 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「ALTER PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「ALTER SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「ALTER TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』
- **BACKUP 文**　以前のリリースでは、TRANSACTION LOG RENAME 句または TRANSACTION LOG TRUNCATE 句で、DBFILE ONLY 句を指定できました。今回のバージョンでは、バックアップには相互に排他的な 2 つの種類があるため、DBFILE ONLY をどちらかの TRANSACTION LOG 句で指定すると、エラーが発生するようになりました。「BACKUP 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **COMMENT 文**　構文 COMMENT ON LOGIN はサポートされなくなりました。代わりに構文 COMMENT ON INTEGRATED LOGIN を使用してください。「COMMENT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **INSERT 文**　SQL Anywhere 10 で ON EXISTING SKIP 句と ON EXISTING ERROR 句を使用するときに、テーブルにデフォルトのカラムが含まれる場合、サーバでは、すでに存在するローに対してもデフォルト値が計算されます。その結果、AUTOINCREMENT などのデフォルト値によって、省略するローにも影響が生じます。AUTOINCREMENT の場合は、AUTOINCREMENT のシーケンスで値が省略されます。以前のバージョンでは、省略するロー

のデフォルトのカラムに対してこれらの計算は行われませんでした。「INSERT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **VALIDATE 文** すべての検証アクティビティ (VALIDATE 文の実行、検証ユーティリティ (dbvalid) の実行など) で、VALIDATE 権限が必要になりました。また、REMOTE DBA パーミッションで検証アクティビティを実行できなくなりました。

  VALIDATE TABLE 文 (および VALIDATE MATERIALIZED VIEW) は、孤立した BLOB を検査します。

  VALIDATE INDEX の区分が変更され、ALTER INDEX 文の構文と一貫性があるようになりました。以前の構文は、サポートはされますが、廃止されます。アプリケーションで VALIDATE INDEX 文を使用している場合は、新しい構文に変更する必要があります。

  これらの変更の詳細については、「VALIDATE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

非推奨機能とサポートされなくなった機能

- **検証ユーティリティ (dbvalid) の -f、-fi、-fd、-fn オプションの廃止** dbvalid ユーティリティの構文が簡略化されました。以前は、オプションを指定しないと、テーブルの検証時に式の検証が実行されていました。今回のバージョンでは、-f、-fi-、-fd オプションを指定したかのように、デフォルトでフル検証が実行されるようになりました。そのため、これらのオプションの使用は廃止され、テーブルで式の検証を実行する場合は -fx オプションを指定する必要があります。

  また、-fn オプション (バージョン 9.0.0 以前のリリースからのアルゴリズムで検証を実行する) はサポートされなくなりました。

  検証ユーティリティの詳細については、「検証ユーティリティ (dbvalid)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **VALIDATE TABLE 文のオプションの廃止** VALIDATE TABLE 文の構文が簡略化されました。以前は、オプションを指定しないと、通常の検証が実行されていました。今回のバージョンでは、WITH FULL CHECK オプションを指定したかのように、デフォルトでフル検証が実行されるようになりました。そのため、WITH FULL CHECK、WITH INDEX、WITH data オプションは廃止されます。「VALIDATE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **Transact-SQL 外部ジョインの廃止** 今回のリリースで Transact-SQL 外部ジョインは廃止され、SQL Anywhere の将来のバージョンではサポートされなくなる予定です。新しい tsql_outer_joins データベース・オプションを使用すると、現在の接続の DML 文とビューで Transact-SQL 外部ジョイン演算子 *= と =* の使用を有効／無効にできます。このオプションのデフォルト設定は Off です。「tsql_outer_joins オプション [互換性]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **WITH HASH SIZE 句のサポート終了** B ツリー・インデックス・テクノロジが削除されたことで、CREATE INDEX 文の WITH HASH SIZE 句はサポートされなくなりました。

- **未サポートのプロパティ**　NumProcessorsAvail と NumProcessorsMax サーバ・プロパティはサポートされなくなりました。代わりに、NumLogicalProcessors、NumLogicalProcessorsUsed、NumPhysicalProcessors、NumPhysicalProcessorsUsed の各サーバ・プロパティを使用できます。「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **LOAD TABLE の STRIP ON 句の廃止**　前後のブランクを削除する機能は SQL Anywhere 10.0.0 で強化され、削除機能を微調整できるようになり、STRIP ON は廃止されます。今後も後続ブランク*だけ*を削除する場合は、代わりに STRIP RTRIM を使用してください。「LOAD TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **UTF8 照合の廃止**　UTF8 照合は廃止されます。代りに UTF8BIN 照合を使用してください。「サポートされている照合と代替照合」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **jConnect 4.5 のサポート終了**　jConnect 4.5 を使用して接続していたアプリケーションは動作しますが、jConnect 5.5 または 6.0.5 を使用することをおすすめします。「jConnect JDBC ドライバの使用」『SQL Anywhere サーバ - プログラミング』を参照してください。
- **SQLLOCALE 環境変数のサポート終了**　SQLLOCALE 環境変数はサポートされなくなりました。SALANG と SACHARSET 環境変数に置き換わっています。「SALANG 環境変数」『SQL Anywhere サーバ - データベース管理』と「SACHARSET 環境変数」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **名前付きパイプのサポート終了**　名前付きパイプはサポートされなくなりました。名前付きパイプを使用していたアプリケーションは、共有メモリを使用するように変更する必要があります。「通信プロトコルの選択」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **データ・ソース ユーティリティ (dbdsn) の -o オプションの廃止**　データ・ソース・ユーティリティの -o オプションは廃止されます。出力メッセージをファイルに書き込む場合は、接続文字列で LogFile 接続パラメータを指定できます。「LogFile 接続パラメータ [LOG]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **カスタム照合の作成の未サポート**　カスタム照合の作成はサポートされなくなりました。[カスタム照合作成] ウィザード、照合ユーティリティ (dbcollat)、DBCollate 関数、a_db_collation 構造体はサポートされなくなりました。「照合の選択」『SQL Anywhere サーバ - データベース管理』を参照してください。

  カスタム照合を使用してデータベースを再構築する場合に 1 ステップで再構築すると、その照合は保存されます。データベースをアンロードしてから、作成したデータベースにスキーマとデータをロードする場合は、提供されるいずれかの照合を使用する必要があります。「バージョン 9 以前のデータベースをバージョン 10.0.0 用に再構築」 360 ページを参照してください。
- **サーバ・ライセンス取得ユーティリティの -p オプションの未サポート**　以前のリリースでは、サーバ・ライセンス取得ユーティリティで -p オプション (データベース・サーバのライセンスが取得されているオペレーティング・システムを指定する) がサポートされていました。このオプションはサポートされなくなりました。

- **データベース・サーバの -d オプションの未サポート** データベース・サーバ・オプション -d (NetWare で DFS (Direct File System) I/O ではなく POSIX I/O を使用する) はサポートされなくなりました。
- **データベース・サーバの -y オプションの未サポート** Windows 95／98／Me がサポートされなくなったため、データベース・サーバ・オプション -y (Windows 95／98／Me で Windows サービスとしてデータベース・サーバを実行する) は、サポートされなくなりました。サポートされるプラットフォームでデータベース・サーバをサービスとして実行するには、dbsvc ユーティリティを使用してください。「Windows 用サービス・ユーティリティ (dbsvc)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **-sc オプションの未サポート** SQL 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。-sc サーバ・オプションを使用すると、SQL 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できます。-sc オプションと C2 サーバ・プロパティのサポートは、バージョン 10.0.0 で削除されました。
- **max_work_table_hash_size データベース・オプションの未サポート** max_work_table_hash_size オプションは、サポートされなくなりました。クエリ・オプティマイザは、テーブル内のデータ分散を基に、内部テンポラリ・テーブル用にハッシュ・サイズを割り当てます。
- **max_hash_size database オプションの未サポート** max_hash_size オプションは、サポートされなくなりました。
- **圧縮データベースとライト・ファイルの未サポート** その結果、次の機能は使用できなくなりました。
  - **ファイル拡張子** 次のファイル拡張子は、サポートされなくなりました。
    - ライト・ファイルを識別するための *.wrt* 拡張子
    - 圧縮データベース・ファイルを識別するための *.cdb* 拡張子
  - **NetWare 上のデータベース・サーバの動作** データベース・サーバは、拡張子なしでデータベース・ファイルが指定されたときに、*.wrt* 拡張子を持つデータベース・ファイルを検索しなくなりました。「SQL Anywhere データベース・サーバ」『SQL Anywhere サーバ - データベース管理』を参照してください。
  - **データベースの読み込み専用メディアへの配備** 読み込み専用メディア (CD-ROM など) で提供されるデータベースへの変更を記録するために、ライト・ファイルを指定できなくなりました。ただし、読み込み専用モードでデータベースが実行されている場合に、読み込み専用メディアでデータベースを配備することはできます。「データベースを読み込み専用メディアで配備する」『SQL Anywhere サーバ - プログラミング』と「-r サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
  - **データベース・ユーティリティ** 次のユーティリティやウィザードは、サポートされなくなりました。
    - [データベース展開] ウィザード
    - [ライト・ファイル作成] ウィザード
    - [データベース展開] ウィザード

- 展開ユーティリティ (dbexpand)
- 圧縮ユーティリティ (dbshrink)
- ライト・ファイル・ユーティリティ (dbwrite)

- **SQL 文**　次の SQL 文は、サポートされなくなりました。

  - ALTER WRITEFILE
  - CREATE WRITEFILE
  - CREATE COMPRESSED DATABASE
  - CREATE EXPANDED DATABASE

- **DBTools 構造体**　次の構造体や構造体のメンバはサポートされなくなりました。

  - **a_backup_db 構造体**　この構造体は、DBTools ライブラリを使用してバックアップ・タスクを実行するために必要な情報を格納します。

    backup_writefile メンバは _unused として表示されるようになりました。

  - **a_compress_db 構造体**　この構造体は削除されました。

  - **a_compress_stats 構造体**　この構造体は、DBTools ライブラリを使用してデータベース圧縮タスクを実行するために必要な情報を格納します。

  - **a_db_info 構造体**　この構造体は、DBTools ライブラリを使用して dbinfo 情報を戻すために必要な情報を格納します。

    wrtbufsize メンバは _unused1 として、wrtnamebuffer メンバは _unused2 として、compressed メンバは _unused3 として表示されるようになりました。

  - **an_expand_db 構造体**　この構造体は、DBTools ライブラリを使用してデータベースを拡張するために必要な情報を格納します。

  - **a_stats_line 構造体**　DBTools ライブラリを使用してデータベースを圧縮と拡張するために必要な情報を格納します。

  - **a_writefile 構造体**　この構造体は、DBTools ライブラリを使用してデータベース・ライト・ファイルを管理するために必要な情報を格納します。

- **DBTools 関数**　次の関数は、サポートされなくなりました。

  - DBChangeWriteFile
  - DBCompress
  - DBCreateWriteFile
  - DBExpand
  - DBStatusWriteFile

- **データベースのプロパティ**　次のデータベース・プロパティは、サポートされなくなりました。

  - Compression
  - FileSize *writefile*
  - FreePages *writefile*

- **DB_BACKUP_WRITEFILE**　この Embedded SQL 関数はサポートされなくなりました。

- **未使用の ASE 互換ビューとプロシージャのサポート終了**　SQL Anywhere データベースで次の未使用の Adaptive Server Enterprise ビュー・サポートは削除されました。

ビュー名	ビュー名
SYSALTERNATES	SYSLOGINROLES
SYSAUDITOPTIONS	SYSLOGS
SYSAUDITS	SYSMESSAGES
SYSCHARSETS	SYSPROCEDURES
SYSCONFIGURES	SYSPROCESSES
SYSCONSTRAINTS	SYSPROTECTS
SYSCURCONFIGS	SYSREFERENCES
SYSDATABASES	SYSREMOTELOGINS
SYSDEPENDS	SYSROLES
SYSDEVICES	SYSSEGMENTS
SYSENGINES	SYSSERVERS
SYSKEYS	SYSSRVROLES
SYSLANGUAGES	SYSTHRESHOLDS
SYSLOCKS	SYSUSAGES

SQL Anywhere データベースで次の未使用の Adaptive Server Enterprise プロシージャ・サポートは削除されました。

プロシージャ名	プロシージャ名
sp_addalias	sp_helpindex
sp_addauditrecord	sp_helpjoins
sp_addlanguage	sp_helpkey
sp_addremotelogin	sp_helplanguage
sp_addsegment	sp_helplog
sp_addserver	sp_helpremotelogin
sp_addthreshold	sp_helpprotect

プロシージャ名	プロシージャ名
sp_adddumpdevice	sp_helpsegment
sp_auditdatabase	sp_helpserver
sp_auditlogin	sp_helpsort
sp_auditobject	sp_helpthreshold
sp_auditoption	sp_helpuser
sp_auditsproc	sp_indsuspect
sp_bindefault	sp_lock
sp_bindmsg	sp_locklogin
sp_bindrule	sp_logdevice
sp_changedbowner	sp_modifylogin
sp_checknames	sp_modifythreshold
sp_checkreswords	sp_monitor
sp_clearstats	sp_placeobject
sp_commonkey	sp_primarykey
sp_configure	sp_procxmode
sp_cursorinfo	sp_recompile
sp_dboption	sp_remap
sp_dbremap	sp_remoteoption
sp_depends	sp_rename
sp_diskdefault	sp_renamedb
sp_displaylogin	sp_reportstats
sp_dropalias	sp_role
sp_dropdevice	sp_serveroption
sp_dropkey	sp_setlangalias
sp_droplanguage	sp_spaceused
sp_dropremotelogin	sp_syntax
sp_dropsegment	sp_unbindefault

プロシージャ名	プロシージャ名
sp_dropserver	sp_unbindmsg
sp_dropthreshold	sp_unbindrule
sp_estspace	sp_volchanged
sp_extendsegment	sp_who
sp_foreignkey	sp_column_privileges
sp_help	sp_databases
sp_helpconstraint	sp_datatype_info
sp_helpdb	sp_server_info
sp_helpdevice	sp_table_privileges
sp_helpgroup	

- **SYSINDEX システム・ビューからの index_type と index_owner カラムの削除**　index_type と index_owner カラムは SYSINDEX ビューから削除されました。これらのカラムは、それぞれデフォルト値 USER と SA が格納されていました。インデックス情報は、ISYSIDX と ISYSIDXCOL システム・ビューに格納されるようになりました。「SYSIDX システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』と「SYSINDEXES システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **サーバでの DLL プロトコル・オプションの削除**　DLL プロトコル・オプションは、Windows 32 ビット・プラットフォームで実行しているクライアントだけに適用されるようになりました。データベース・サーバでは、DLL プロトコル・オプションが削除され、Winsock 2.2 だけを使用します。同様に、Windows CE クライアントでは、DLL プロトコル・オプションが削除され、Winsock 1.1 だけを使用します。

  Winsock 2.2 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。「DLL プロトコル・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **ASANY と ASANYSH 環境変数名の変更**　ASANY と ASANYSH 環境変数は、それぞれ SQLANY10 と SQLANYSH10 に名前が変更されました。「SQLANY10 環境変数」『SQL Anywhere サーバ - データベース管理』と「SQLANYSAMP10 環境変数」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **PreserveSource プロパティの廃止**　今回のリリースで PreserveSource データベース・プロパティが廃止され、この設定が問い合わされると常に値 On を返すようになりました。

- **サポートされなくなったデータベース・プロパティ**　今回のリリースでは、次のデータベース・プロパティが削除されました。

  - BlobArenas
  - ClusteredIndexes

- CompressedBTrees
- FileVersion
- FreePageBitMaps
- Histograms
- HistogramHashFix
- IndexStatistics
- LargeProcedureIDs
- NamedConstraints
- SeparateCheckpointLog
- SeparateForeignKeys
- StringHistogramsFix
- TableBitMaps
- TablesQualTriggers
- TransactionsSpanLogs
- UniqueIdentifier
- VariableHashSize

- **サポートされなくなったシステム・プロシージャ**　sa_conn_properties_by_name と sa_conn_properties_by_conn システム・プロシージャはサポートされなくなりました。新しい sa_conn_options システム・プロシージャを使用すると、同じ情報を取得できます。「sa_conn_options システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **クエリ最適化プランから削除されたアルゴリズム**　ロック、ネスト・ブロック・ジョイン、ソート・ブロック、JNBO の各アルゴリズムは、クエリ最適化プランから削除されました。また、ロック・ノードはこのプランに表示されなくなりました。ロック情報は、プラン内のスキャン・ノードで確認できます。

- **util_db.ini ファイルの廃止**　ユーティリティ・データベースへの接続時に、*util_db.ini* ファイルを使用して DBA ユーザのパスワードを指定する機能は廃止されました。代わりに -su サーバ・オプションを使用できます。「ネットワーク・データベース・サーバでの util_db.ini の使用 (旧式)」『SQL Anywhere サーバ - データベース管理』と「-su サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **廃止された Windows CE プラットフォーム**　Windows CE MIPS プロセッサのサポートは削除されました。サポートされるプラットフォームのリストについては、「サポートされるプラットフォーム」『SQL Anywhere 10 - 紹介』を参照してください。

Mobile Link

次の項では、Mobile Link のバージョン 10.0.0 での新機能、動作の変更、廃止される機能について説明します。

新機能

次に、バージョン 10.0.0 で導入された Mobile Link の追加機能を示します。

Mobile Link の主な新機能

Sybase Central の Mobile Link プラグインの強化

同期モデル・ファイルの作成にウィザードを使用することにより、Mobile Link アプリケーションの設定が大幅に簡単になりました。このファイルには、リモート・テーブルと統合化テーブルについて入力する情報と、それらを同期する方法についての情報が含まれます。モデルが用意できたら、別のウィザードを使用して展開します。展開されたモデルによってアプリケーションに必要なスクリプトとテーブルが生成されます。

- **[同期モデル作成] ウィザード**　新しい [同期モデル作成] ウィザードを使用すると、Mobile Link アプリケーションをすばやく作成して展開できます。このウィザードではリモート・データベースを作成することも、既存のリモート・データベースを使用することもできます。同期スクリプトの作成が自動化されるほか、ダウンロードの削除、競合の解決、その他の難しい同期の問題が自動的に処理されます。

  「[同期モデル作成] ウィザード」 『Mobile Link - クイック・スタート』を参照してください。

- **モデル・モード**　[同期モデル作成] ウィザードを使用したあとは、モデル・モードを使用して、同期プロジェクトを展開前にカスタマイズします。モデル・モードでは、オフラインで作業します。モデル・モードでは、同期モデルを XML ファイルとして格納します。

  「モデル・モード」 『Mobile Link - クイック・スタート』を参照してください。

- **[展開] ウィザード**　モデルをカスタマイズしたら、新しい [展開] ウィザードを使用してモデルを展開できます。[展開] ウィザードでは、スクリプト、ユーザ、スクリプトのバージョンなどを統合データベース上の Mobile Link システム・テーブルに追加します。統合データベースに加えた変更は、モデル・モードにリエンジニアリングできませんが、同じモデルを複数回展開できます。

  「モデルの配備」 『Mobile Link - クイック・スタート』を参照してください。

- **管理モード**　バージョン 10.0.0 以前に存在していた Mobile Link プラグインは、管理モードと呼ばれるようになりました。使いやすくなるように、管理モードにはさまざまな強化が行われています。管理モードでは、統合データベースに接続されており、変更はすぐに有効になります。管理モードを使用すると、Mobile Link 統合データベースのすべてを変更できます。

  「管理モード」 『Mobile Link - クイック・スタート』を参照してください。

任意のデータ・ソースとの同期

「**ダイレクト・ロー・ハンドリング**」と呼ばれる新機能を使用すると、ほとんどすべてのデータソースと同期できます。たとえば、SQL ベースのロー・ハンドリング (MySQL など) を使用して、アプリケーション・サーバ、Web サーバ、Web サービス、テキスト・ファイル、Excel スプレッドシート、J2ME デバイス、または統合データベースとして使用できない RDBMS と同期できます。ただし、Mobile Link システム・テーブルや、Mobile Link で管理するデータを保持するために、統合データベースは必要なままです。新しいデータ・ソースは、同期処理に完全に統合させることができます。

Mobile Link サーバの API が拡張され、ダイレクト・ロー・ハンドリングがサポートされるようになりました。さらに、2 つの新しいイベントが追加されました。次の項を参照してください。

- 「ダイレクト・ロー・ハンドリング」『Mobile Link - サーバ管理』
- 「handle_DownloadData 接続イベント」『Mobile Link - サーバ管理』
- 「handle_UploadData 接続イベント」『Mobile Link - サーバ管理』

また、「**モバイル Web サービス**」と呼ばれる新しい機能では、モバイルに最適化された、リモート・アプリケーションと統合可能な非同期 Web サービスを提供します。

「モバイル Web サービス」『QAnywhere』を参照してください。

配備の簡略化

Mobile Link サーバ、SQL Anywhere クライアント、Mobile Link モニタ、暗号化コンポーネントを配備するための [配備] ウィザードが使用できるようになりました。以前のバージョンで提供されていた InstallShield マージ・モジュールとテンプレートは、提供されなくなりました。次の項を参照してください。

- 「配備ウィザードの使用」『SQL Anywhere サーバ - プログラミング』
- 「Mobile Link アプリケーションの配備」『Mobile Link - サーバ管理』

統合データベース

新しい設定プロシージャ

- **SQL Anywhere 統合データベースで必要な設定スクリプト**　Mobile Link 統合データベースとして SQL Anywhere データベースを使用する前に、設定スクリプトを実行する必要があります。また、設定スクリプトで作成された Mobile Link システム・テーブルは、設定スクリプトを実行したユーザが所有者になりました。この動作は、他の種類の統合データベースと同じです。以前のバージョンの Mobile Link では、Mobile Link システム・テーブルの所有者は SQL Anywhere 統合データベースの DBO でした。

  「SQL Anywhere 統合データベース」『Mobile Link - サーバ管理』を参照してください。

- **単純化された設定プロシージャ**　統合データベースは、Sybase Central で設定したり、設定スクリプトを使用して設定したりできるようになりました。以前は、設定スクリプトを実行する必要がありました。また、統合データベースの種類ごとに、設定スクリプトは 1 つだけになりました。バージョン固有の設定スクリプト (*syncase125.sql* など) は不要です。

「Mobile Link 統合データベース」 『Mobile Link - サーバ管理』を参照してください。

新しいシステム・オブジェクト

- **統合データベース上の Mobile Link システム・テーブルをクリーンアップする新しい方法** 次の処理を実行できる新しいシステム・プロシージャが追加されました。
  - Mobile Link システム・テーブルからの旧式のリモート・データベースの情報の消去。

    「ml_delete_sync_state_before」 『Mobile Link - サーバ管理』を参照してください。
  - 未使用または不要の同期ステータス情報の削除。

    「ml_delete_sync_state」 『Mobile Link - サーバ管理』を参照してください。
  - 同期ステータス情報のリセット。

    「ml_reset_sync_state」 『Mobile Link - サーバ管理』を参照してください。
- **新しい Mobile Link サーバ・システム・テーブルとスキーマ** Mobile Link システム・テーブルは、次のように変更されています。
  - 複数の新しい Mobile Link システム・テーブルが追加されました。次の項を参照してください。
    - 「ml_database」 『Mobile Link - サーバ管理』
    - 「ml_column」 『Mobile Link - サーバ管理』
    - 「ml_qa_clients」 『Mobile Link - サーバ管理』
  - ml_subscription の内容が大幅に変わりました。Ultra Light 同期シーケンス番号は、以前は ml_user.commmit_state に格納されていましたが、ml_subscription.progress に格納されるようになりました。progress カラムには、SQL Anywhere リモート同期の進行状況も格納されます。
    「ml_subscription」 『Mobile Link - サーバ管理』を参照してください。
  - ml_user の内容が大幅に変わりました。
    「ml_user」 『Mobile Link - サーバ管理』を参照してください。
  - checksum カラムが ml_script テーブルに追加されました。
    「ml_script」 『Mobile Link - サーバ管理』を参照してください。
  - ml_listening の ml_user カラムが name カラムに変更されました。
    「ml_listening」 『Mobile Link - サーバ管理』を参照してください。
  - サーバ起動同期のために Sybase Central で内部的に使用される新しいシステム・テーブルが追加されました。
    「ml_sis_sync_state」 『Mobile Link - サーバ管理』を参照してください。
  - 複数の QAnywhere システム・テーブルに変更が加えられました。次の項を参照してください。
    - 「ml_qa_delivery」 『Mobile Link - サーバ管理』
    - 「ml_qa_delivery_client」 『Mobile Link - サーバ管理』

- 「ml_qa_global_props」『Mobile Link - サーバ管理』
- 「ml_qa_global_props_client」『Mobile Link - サーバ管理』
- 「ml_qa_repository」『Mobile Link - サーバ管理』
- 「ml_qa_repository_client」『Mobile Link - サーバ管理』
- 「ml_qa_repository_props」『Mobile Link - サーバ管理』
- 「ml_qa_repository_props_client」『Mobile Link - サーバ管理』
- 「ml_qa_repository_staging」『Mobile Link - サーバ管理』

- **新しいシステム・プロシージャ ml_add_column** 名前付きロー・パラメータを使用するときに、場合によっては、この新しいシステム・プロシージャを使用して、ml_column Mobile Link システム・テーブルにリモート・データベースのカラムの情報を移植する必要があります。

  「ml_add_column」『Mobile Link - サーバ管理』を参照してください。

Mobile Link サーバ

mlsrv10 の新機能

- **サーバ名を mlsrv10 に変更** Mobile Link サーバは mlsrv10 という名前になりました。以前は、dbmlsrv9 と呼ばれていました。

- **mlsrv10 -x の新しい構文** mlsrv10 -x オプション (Mobile Link クライアントのネットワーク・プロトコル・オプションを設定) が変更されました。

  「-x オプション」『Mobile Link - サーバ管理』を参照してください。

- **古いクライアント用の新しい -xo オプション** Mobile Link サーバをバージョン 8 または 9 クライアントに接続するには、mlsrv10 -xo オプションを使用する必要があります。このオプションは、dbmlsrv9 -x オプションと等価です。バージョン 10 のクライアントと同様に、mlsrv10 の 1 インスタンスからバージョン 8 と 9 のクライアントをサポートできます。ただし、それには 2 つの異なるポートを使用する必要があります。

  HTTP と HTTPS の -xo オプションに新しいオプション session_key が含まれます。このオプションは、接続の追跡に JSESSIONID を使用できない場合に便利です。

  「-xo オプション」『Mobile Link - サーバ管理』を参照してください。

- **キャッシュ処理の向上** Mobile Link サーバは、タスクごとに別のメモリ・プールを管理しなくなりました。すべてのキャッシュ・メモリはすべての同期で共有されます。キャッシュ・サイズを設定するには、新しい mlsrv10 -cm オプションを使用します。キャッシュ・サイズを設定するその他のオプション (-bc、-d、-dd、-u) は削除されました。

  「-cm オプション」『Mobile Link - サーバ管理』を参照してください。

- **すべてのストリームに影響するようになった ignore オプション** 無視するように指定したすべてのホスト名または IP アドレスが、すべての -x ストリームで無視されるようになりました。以前は (-xo を使用した場合は今回のバージョンでも)、ホストは指定されたストリームでのみ無視されていました。

  「-x オプション」『Mobile Link - サーバ管理』の「ignore」を参照してください。

- **スクリプトの強制アップロード**　mlsrv10 -zus オプションを使用すると、アップロードするデータがテーブルにない場合でも、Mobile Link サーバでそのテーブルのアップロード・スクリプトを強制的に呼び出すことができます。

  「-zus オプション」『Mobile Link - サーバ管理』を参照してください。

- **新しい冗長性オプション**　新しい冗長性オプション **e** を使用すると、システム・イベント・スクリプトを取得できます。-ve を指定すると、Mobile Link サーバでは、Mobile Link システム・テーブルを管理するためのすべてのシステム・イベント・スクリプトと、アップロード・ストリームを定義する SQL 文を表示します。

  「-v オプション」『Mobile Link - サーバ管理』を参照してください。

- **ファイル転送ディレクトリ**　ファイル転送用のディレクトリを使用できる新しいオプションが追加されました。

  「-ftr オプション」『Mobile Link - サーバ管理』を参照してください。

- **同時同期処理の最大数の設定**　アクティブに作業できる同期処理の最大数を設定することで、パフォーマンスを向上できるようになりました。

  「-sm オプション」『Mobile Link - サーバ管理』を参照してください。

- **同時ネットワーク接続の制限**　新しい -nc オプションを使用すると、同時ネットワーク接続の数の上限を指定できます。

  「-nc オプション」『Mobile Link - サーバ管理』を参照してください。

- **mlsrv10 によるメッセージのタイムスタンプへの ISO 8601 日時フォーマットの使用**　情報、警告、エラーの各メッセージのタイムスタンプは、明確に定義された ISO 8601 日時フォーマット ({**I**|**W**|**E**}.*yyyy-mm-dd hh:mm:ss message*) を使用するようになりました。

Mobile Link の新しいスクリプト機能

- **名前付きのスクリプト・パラメータ**　Mobile Link イベント・パラメータに名前が付けられました。以前は、疑問符でスクリプト・パラメータを指定する必要がありました。今回のバージョンでは、疑問符はオプションになりました。定義済みの名前付きパラメータのセットから選択したり、独自の名前付きパラメータを作成したりすることができます。RDBMS で変数をサポートしない場合は、ユーザ定義の名前付きパラメータが便利です。疑問符の場合とは異なり、名前付きパラメータは任意の順序で指定でき、使用できるパラメータの任意のサブセットを使用できます。また、ほとんどの場合で同じ名前付きパラメータを同じスクリプト内で複数回使用できます。

  「スクリプトのパラメータ」『Mobile Link - サーバ管理』を参照してください。

- **新しい競合検出イベント**　カラム・レベルで競合を検出するためにスクリプト化できる新しいイベントがあります。このイベントは、ロー・レベルで競合を検出する upload_fetch イベントの代わりになるものです。

  「upload_fetch_column_conflict テーブル・イベント」『Mobile Link - サーバ管理』を参照してください。

- **グローバル・スクリプト・バージョン**　グローバル・スクリプト・バージョンを作成できるようになりました。グローバル・スクリプト・バージョンに関連するスクリプトを定義する

と、同期に使用しているスクリプト・バージョンで同じイベントのスクリプトを指定しない場合は、すべての同期処理でそのスクリプトが自動的に使用されます。つまり、複数のスクリプト・バージョンを使用している場合は、接続レベル・スクリプトの複製を避けることができるということです。

「ml_global スクリプト・バージョン」 『Mobile Link - サーバ管理』を参照してください。

パフォーマンスの強化

- **Mobile Link アーキテクチャの向上** Mobile Link サーバはアーキテクチャが再構成され、スループット、柔軟性、保守性が向上しました。同様の理由で、内部の Mobile Link クライアント／サーバ・プロトコルも強化されました。

  「Mobile Link のパフォーマンス」 『Mobile Link - サーバ管理』を参照してください。

サーバのその他の強化

- **スナップショット・アイソレーション** SQL Anywhere バージョン 10 と Microsoft SQL Server 2005 以降の統合データベースでは、スナップショット・アイソレーションがダウンロードのデフォルト、アップロードのオプションになりました。この動作を制御できるように、Mobile Link サーバのオプションが追加されました。

  次の項を参照してください。

  - 「Mobile Link 独立性レベル」 『Mobile Link - サーバ管理』
  - 「-dsd オプション」 『Mobile Link - サーバ管理』
  - 「-dt オプション」 『Mobile Link - サーバ管理』
  - 「-esu オプション」 『Mobile Link - サーバ管理』

- **同期 ID** 各同期が 1 ー 4294967295 の間の整数で識別されるようになりました。Mobile Link サーバの各インスタンスは、独自の同期 ID を管理します。Mobile Link サーバが起動すると、ID は 1 にリセットされます。この ID は出力ファイルにロギングされます。

- **Mobile Link ネットワーク・レイヤの向上** ネットワーク・レイヤには、圧縮、永続的な接続 (同じ接続で複数回の同期が可能)、IPv6 サポートなどが含まれるようになり、エラー検出、活性検出、デバッグが向上しました。

Mobile Link モニタの強化

- **ユーティリティ名を mlmon に変更** Mobile Link モニタは mlmon という名前になりました。以前は、dbmlmon と呼ばれていました。

  「Mobile Link モニタの起動」 『Mobile Link - サーバ管理』を参照してください。

- **複数の Mobile Link モニタ** 複数の Mobile Link モニタを同じ Mobile Link サーバに同時に接続できるようになりました。これによって、複数のユーザが同じサーバで同期処理を追跡できます。

  「Mobile Link モニタの起動」 『Mobile Link - サーバ管理』を参照してください。

- **ネットワーク・オプション** Mobile Link モニタで、Mobile Link クライアントと同じネットワーク・オプションを使用できるようになりました。

「Mobile Link モニタの起動」『Mobile Link - サーバ管理』を参照してください。

- **新しい使用率グラフ** [使用率グラフ] ウィンドウ枠には、Mobile Link サーバ内のキューの長さが表示されます。

  「[使用率グラフ] ウィンドウ枠」『Mobile Link - サーバ管理』を参照してください。

- **[チャート] ウィンドウ枠でデータの表示** [チャート] ウィンドウ枠では、従来どおりユーザごとのデータを表示することができるほか、コンパクト・ビューでデータを表示することもできます。コンパクト・ビューでは、できるだけ少ないローでアクティブな同期処理をすべて表示します。同期が単一のワーカ・スレッドに関連付けされなくなったので、ワーカ・ビューは削除されました。

  「[チャート] ウィンドウ枠」『Mobile Link - サーバ管理』を参照してください。

- **新しい [サンプル・プロパティ] ダイアログ** 新しい [サンプル・プロパティ] には、1 秒間隔のデータ、または選択した期間における 1 秒間隔のすべての平均が表示されます。

  「サンプル・プロパティ」『Mobile Link - サーバ管理』を参照してください。

- **強化された [セッション・プロパティ] ダイアログ** セッション・プロパティには、詳細な [統計] タブが含まれるようになりました。

  「セッション・プロパティ」『Mobile Link - サーバ管理』を参照してください。

- **FIPS 対応サーバのモニタ機能** Mobile Link モニタは、FIPS 承認の暗号化を実行している Mobile Link サーバをモニタできるようになりました。以前は、この機能はありませんでした。

- **統計プロパティの変更** 「Mobile Link サーバの変更」 112 ページの「統計プロパティの変更」を参照してください。

Mobile Link リダイレクタの強化

- **リダイレクタで Mobile Link サーバのグループをサポート** 一部のリダイレクタで、Mobile Link サーバ・グループを作成できるようになりました。サーバ・グループは、1 つのリダイレクタ経由でバージョン 8 または 9 クライアントと同時にバージョン 10 クライアントをサポートするため、またはその他の目的で使用できます。各サーバ・グループをサポートするリダイレクタの詳細については、「サポートされるプラットフォーム」『SQL Anywhere 10 - 紹介』を参照してください。

  「Mobile Link サーバ・グループ」『Mobile Link - サーバ管理』を参照してください。

  サーバ・グループをサポートしているリダイレクタは、設定も強化されています。また、*redirector_server_group.config* という名前の新しいサンプル設定ファイルが使用されます。

  「リダイレクタのプロパティの設定 (サーバ・グループをサポートするリダイレクタの場合)」『Mobile Link - サーバ管理』を参照してください。

- **HTTPS のサポート** 以前のバージョンの SQL Anywhere では、リモート・データベースと Web サーバの間の通信で HTTPS を使用する場合、Web サーバで HTTPS が復号化されて、HTTP がリダイレクタ経由で Mobile Link に送信されます。今回のバージョンでは、一部の Web サーバで、ストリームがリダイレクタで HTTPS に再暗号化されてから、Mobile Link サーバに送信されるようになりました。リダイレクタの設定ファイルでは、ML ディレクティブ用の新

しい構文があります。HTTPS をサポートする Web サーバの詳細については、「サポートされるプラットフォーム」『SQL Anywhere 10 - 紹介』を参照してください。

「リダイレクタのプロパティの設定 (サーバ・グループをサポートするリダイレクタの場合)」『Mobile Link - サーバ管理』を参照してください。

UNIX／Linux の強化

- **コンソール**　Linux のインストール環境では、dbmlsync と mlsrv10 のログ情報を表示する GUI コンソールが用意されました。

  「-ux オプション」『Mobile Link - クライアント管理』と「-ux オプション」『Mobile Link - サーバ管理』を参照してください。

- **より一貫性のある文字変換**　UNIX／Linux と Windows の間の文字変換の一貫性が向上しました。

Mobile Link クライアント

- **新しいリモート ID**　Mobile Link は、リモート ID と呼ばれる新しい識別子を使用して、リモート・データベースをユニークに識別するようになりました。以前のバージョンでは、Mobile Link のユーザ名が使用されていました。リモート ID はリモート・データベースに格納されます。Mobile Link は、リモート・データベースが初めて同期されるとき (または、NULL 値のリモート ID が出現したとき) に、リモート ID を生成します。リモート ID は GUID として自動的に作成されますが、自分にとってわかりやすい文字列に設定することもできます。リモート ID によって、同じ Mobile Link ユーザが複数のリモート・データベースを同期できるようになります。Ultra Light リモート・データベースでは、リモート ID を使用することで、複数の Mobile Link ユーザが同じリモート・データベースを同期できるようになります。

  Mobile Link ユーザ名をパラメータとして受け入れる各スクリプトで、remote_id パラメータも受け入れるようになりました。remote_id パラメータは、名前付きパラメータを使用する場合にかぎり使用できます。

  リモート ID を変更できるように、新しいデータベース・オプション ml_remote_id が SQL Anywhere と Ultra Light データベースの両方に追加されました。

  次の項を参照してください。

  - 「リモート ID」『Mobile Link - クライアント管理』
  - 「Mobile Link ユーザ名とリモート ID」 119 ページ
  - SQL Anywhere クライアント：「リモート ID の設定」『Mobile Link - クライアント管理』
  - Ultra Light クライアント：「Ultra Light ml_remote_id オプション」『Ultra Light - データベース管理とリファレンス』

- **新しいファイル転送機能**　データの同期に使用しているのと同じネットワーク・パスを使用して、リモート・デバイスにファイルを転送できる新しい機能が用意されました。SQL Anywhere クライアントでは mlfiletransfer ユーティリティ、Ultra Light クライアントでは新しい MLFileTransfer メソッドを使用できます。この機能は、新しいリモート・データベースにデータを設定する場合や、ソフトウェアをアップグレードする場合に特に便利です。必要に応じて

ファイル転送を認証するための新しい Mobile Link イベントが追加されました。次の項を参照してください。

- SQL Anywhere クライアント：「Mobile Link ファイル転送ユーティリティ [mlfiletransfer]」『Mobile Link - クライアント管理』
- Ultra Light クライアント：「Mobile Link ファイル転送の使用」『Mobile Link - クライアント管理』
- Mobile Link サーバ：「authenticate_file_transfer 接続イベント」『Mobile Link - サーバ管理』

- **SendColumnNames の変更** 以前は、dbmlsync 拡張オプション SendColumnNames と Ultra Light 同期パラメータ Send Column Names は、Mobile Link サーバがサンプル同期スクリプトを生成できるように、リモート・データベースのカラム情報をアップロードするのに使用されていました。サンプル同期スクリプトの作成機能は削除されました ([同期モデル作成] ウィザードに置き換わりました)。SendColumnNames は、ダイレクト・ロー・ハンドリングでのみ使用されるようになりました。次の項を参照してください。

  - 「ダイレクト・ロー・ハンドリング」『Mobile Link - サーバ管理』
  - 「SendColumnNames (scn) 拡張オプション」『Mobile Link - クライアント管理』
  - 「Send Column Names 同期パラメータ」『Mobile Link - クライアント管理』

- **単純化された活性タイムアウト設定** 活性タイムアウトをクライアントで制御できるようになりました。新しいネットワーク・プロトコル・オプション timeout が導入されました。このオプションは liveness_timeout、contd_timeout、unknown_timeout、network_connect_timeout を置き換えます。

  「timeout」『Mobile Link - クライアント管理』を参照してください。

- **Buffer_size の強化** buffer_size ネットワーク・プロトコル・オプションを使用して、TCP/IP プロトコルの書き込みバッファ処理や HTTP プロトコルの HTTP 本文サイズを制御できるようになりました。デフォルト値も変更されています。

  「buffer_size」『Mobile Link - クライアント管理』を参照してください。

Ultra Light クライアント

- **Palm での network_leave_open のサポート** Palm デバイスで、同期の完了後にネットワーク接続を開いたままにするかどうかを選択できるようになりました。以前のリリースでは、他のプラットフォームでこの機能を使用できていました。

  「network_leave_open」『Mobile Link - クライアント管理』を参照してください。

- **Ultra Light の強化** Ultra Light の強化については、「同期」 136 ページを参照してください。

SQL Anywhere クライアント

- **スクリプト化されたアップロード** 通常の同期では、dbmlsync はトランザクション・ログを使用してアップロードを作成し、前回のアップロード後にリモート・データベースで変更されたすべての関連データを同期します。アップロードされるローを正確に定義するストアド・プロシージャを記述し、トランザクション・ログの使用をバイパスできるようになりました。これらのストアド・プロシージャは DML を実行して結果セットをアップロードできるため、必要に応じてローを動的に作成できます。

「スクリプト化されたアップロード」『Mobile Link - クライアント管理』を参照してください。

スクリプト化されたアップロードをサポートするために、SQL Anywhere システム・オブジェクトは次のように変更されました。

- ISYSPUBLICATION システム・テーブルに新しいカラム (sync_type) が追加されました。「SYSPUBLICATION システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- ISYSSYNCSCRIPT システム・テーブルに、同期スクリプトを追跡するための新しいカタログ・オブジェクトが追加されました。

  「SYSSYNCSCRIPT システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- 新しいシステム・プロシージャ変換進行状況値が追加されました。次の項を参照してください。
  - 「sa_convert_ml_progress_to_timestamp システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』
  - 「sa_convert_timestamp_to_ml_progress システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』

- **dbmlsync の新しいスケジュール・オプション**　EVERY と INFINITE スケジュール・オプションを使用するときに、dbmlsync の開始時に同期を実行しないことを指定できるようになりました。

  「NoSyncOnStartup (nss) 拡張オプション」『Mobile Link - クライアント管理』を参照してください。
- **ダウンロード専用のパブリケーション**　データをダウンロードするだけのパブリケーションを作成できるようになりました。ダウンロード専用のパブリケーションでは、ログ・ファイルを使用しません。

  「ダウンロード専用のパブリケーション」『Mobile Link - クライアント管理』を参照してください。
- **エラー処理の強化**　新しいイベント・フックが追加され、クライアントで dbmlsync がレポートするエラーを処理できるようになりました。

  次の項を参照してください。
  - 「イベント・フック・プロシージャ内でのエラーと警告の処理」『Mobile Link - クライアント管理』
  - 「sp_hook_dbmlsync_all_error」『Mobile Link - クライアント管理』
  - 「sp_hook_dbmlsync_communication_error」『Mobile Link - クライアント管理』
  - 「sp_hook_dbmlsync_misc_error」『Mobile Link - クライアント管理』
  - 「sp_hook_dbmlsync_sql_error」『Mobile Link - クライアント管理』

- **dbmlsync でテーブル順序の確保の停止**　デフォルトでは、子テーブルが親テーブルより先にアップロードされると、dbmlsync はエラーを発行します。新しい拡張オプションを使用すると、この動作を上書きできます。

  「TableOrderChecking (toc) 拡張オプション」『Mobile Link - クライアント管理』を参照してください。

- **永続的な接続**　複数の同期間で Mobile Link サーバへの接続を開いたままにするように、dbmlsync に指定できるようになりました。

  「-pc オプション」『Mobile Link - クライアント管理』を参照してください。

- **同期を追跡する新しい方法**　SQL Anywhere リモート・データベースの場合のみ、begin_publication または end_publication スクリプトで subscription_id パラメータを指定できるようになりました。この値は、SYSSYNC システム・テーブルで sync_id と呼ばれます。同期の情報を追跡できる高度な機能です。次の項を参照してください。

  - 「begin_publication 接続イベント」『Mobile Link - サーバ管理』
  - 「end_publication 接続イベント」『Mobile Link - サーバ管理』

- **dbmlsync によるメッセージのタイムスタンプへの ISO 8601 日時フォーマットの使用**　情報、警告、エラーの各メッセージのタイムスタンプは、明確に定義された ISO 8601 日時フォーマット ({**I**|**W**|**E**}.*yyyy-mm-dd hh:mm:ss message*) を使用するようになりました。

- **#hook_dict で拡張された値**　dbmlsync ユーティリティは、フックを公開し、テンポラリ・テーブル #hook_dict に名前／値ペアを値として渡します。以前は、#hook_dict テーブル内の値は VARCHAR (255) として定義されていました。これが VARCHAR (10240) に増加しました。

セキュリティ

- **RSA が SQL Anywhere に付属**　FIPS を使用していない場合は、RSA 暗号化を使用するためのライセンスを別に購入する必要がなくなりました。

- **HTTPS で使用可能な ECC 暗号化**　HTTPS の使用時に、ECC 暗号化を使用できるようになりました。

- **新しい mlsrv10 -fips オプション**　Mobile Link サーバの起動時に -fips を指定して、すべてのセキュア接続で FIPS 承認のアルゴリズムを使用させることができるようになりました。この設定は、セキュアでないストリームには影響しません。

  「-fips オプション」『Mobile Link - サーバ管理』を参照してください。

- **新しい mluser -fips オプション**　Mobile Link ユーザ認証ユーティリティで、FIPS セキュリティの使用を強制するオプションが提供されるようになりました。

  「Mobile Link ユーザ認証ユーティリティ [mluser]」『Mobile Link - サーバ管理』を参照してください。

- **複数のプラットフォームで FIPS セキュリティのサポート**　複数のプラットフォームで FIPS セキュリティがサポートされるようになりました。サポートされるプラットフォームのリスト

については、「サポートされるプラットフォーム」『SQL Anywhere 10 - 紹介』を参照してください。

- **単純化されたセキュリティ・ストリームの指定方法** セキュリティ・オプションを個別のネットワーク・プロトコルとして扱うことにより、セキュリティ・オプションを指定する構文がサーバとクライアントの両方で単純化されました。サポートされるプロトコルは TCP/IP、TLS (TLS セキュリティを使用した TCP/IP 上の同期)、HTTP、HTTPS になりました。Ultra Light セキュリティ・パラメータは削除されました。

  次の項を参照してください。

  - Mobile Link サーバ :「-x オプション」『Mobile Link - サーバ管理』
  - Mobile Link クライアント :「Mobile Link クライアント・ネットワーク・プロトコル・オプション」『Mobile Link - クライアント管理』

- **オペレーティング・システムとの統合** デフォルトで Mobile Link クライアントは、動作するオペレーティング・システムで信頼済みの証明書を信頼します。

サーバ起動同期

使いやすさ

- **サーバ起動同期は設定が大幅に簡易化されました。** サーバ起動同期アプリケーションの設定が非常に早くなるような強化が加えられました。

  - **Sybase Central のサポート** Notifier と Listener を Sybase Central モデル・モードで設定できるようになり、便利な通知サービスのサブセットを使用できるようになりました。モデル・モードでは、サーバ起動同期のテーブルを指定すると、通知目的で使用されるデータを判断するために download_cursor が自動的に使用されます。ダウンロード・カーソルの変更でデータが識別されると、通知が送信されます。[配備] ウィザードは、対応する Listener オプション・ファイルを生成します。

    「モデル・モードでのサーバ起動同期の設定」『Mobile Link - クイック・スタート』を参照してください。

  - **新しいデフォルト・ゲートウェイ** 新しいゲートウェイ SYNC ゲートウェイを使用すると、Mobile Link に使用するのと同じ種類の通信パス上で永続的な接続を確立できます。SYNC ゲートウェイは、デフォルトのデバイス・トラッカ・ゲートウェイになりました。通知はまず SYNC ゲートウェイを試行し、UDP ゲートウェイ、SMTP ゲートウェイの順にフォールバックします。

    「ゲートウェイと Carrier」『Mobile Link - サーバ起動同期』を参照してください。

Notifier の強化

- **共有接続** 複数の Notifier で、同じデータベースを共有できるようになりました。これにより、統合データベースでの競合と必須サーバ・リソースが低減します。

  「shared_database_connection プロパティ」『Mobile Link - サーバ起動同期』を参照してください。

- **Notifier でのリモート・デバイスの文字セットの使用**　リモート・デバイスの文字セットを使用して、通知がリモート・デバイスに送信されるようになりました。デバイス・トラッキング情報は、統合データベースへの適用前に変換されます。
- **カスタム確認処理**　SQL で、Push 要求の確認を処理してそのステータスを返す Notifier プロパティを実装できるようになりました。

  「confirmation_handler プロパティ」『Mobile Link - サーバ起動同期』を参照してください。
- **カスタム・エラー処理**　SQL で、Push 要求が未配信、未確認、不適切に確認されたときなどにエラーを処理する Notifier プロパティを実装できるようになりました。

  「error_handler プロパティ」『Mobile Link - サーバ起動同期』を参照してください。

リスナの強化

- **永続的な接続**　Windows Listener で、永続的な接続をサポートするようになりました。デフォルトで Listener は、デバイスの追跡、通知、確認のために、Mobile Link サーバへの永続的な接続を管理するようになりました。この機能により、以前のバージョンと比べて大幅にパフォーマンスが強化されました。dblsn -pc オプションを使用すると、この機能を無効にできます。

  「Listener 構文」『Mobile Link - サーバ起動同期』を参照してください。
- **新しい／変更された Windows Listener オプション**　Listener で次のオプションがサポートされるようになりました。

オプション	説明
-ni	-x を使用する場合に UDP アドレスのトラッキングを停止する。以前は -g と呼ばれていた。
-pc{**+**|**-**}	通知用に永続的な接続を有効／無効にする。
-ns	Windows Mobile 2003 以降の Phone Edition で、デフォルトの SMS 受信を無効にする。
-nu	デフォルトの UDP 受信を無効にする。
-r	$remote_id 変数で使用するリモート ID ファイルを登録する。
-v	1 以上に設定すると、冗長性オプションによってコマンド・ライン・オプションが表示され、ログに記録される。

- **リモート ID ファイル**　Listener コマンド・ラインで、リモート ID ファイルを使用して新しい Mobile Link リモート ID (デフォルトは GUID) にアクセスできるようになりました。これを行うには、新しい dblsn オプションの -r と、新しい Listener action 変数 $remote_id を使用します。

  「Listener 構文」『Mobile Link - サーバ起動同期』と「action 変数」『Mobile Link - サーバ起動同期』を参照してください。

- **認証用の新しい Listener action 変数** メッセージ・ハンドラで使用すると便利な新しい action 変数 $ml_user と $ml_password が追加されました。

  「action 変数」『Mobile Link - サーバ起動同期』を参照してください。

- **接続パラメータ用の新しい Listener action 変数** 新しい $ml_connect action 変数が、これまで dblsn -x オプションを使用して指定していた Mobile Link 接続パラメータに拡張されました。

  「action 変数」『Mobile Link - サーバ起動同期』を参照してください。

- **Listener によるメッセージのタイムスタンプへの ISO 8601 日時フォーマットの使用** 情報、警告、エラーの各メッセージのタイムスタンプは、明確に定義された ISO 8601 日時フォーマット ({**I|W|E**}.*yyyy-mm-dd hh:mm:ss message*) を使用するようになりました。

- **Listener での TLS の使用** Listener は、他の Mobile Link クライアントと同様に、あらゆるネットワーク・オプションを使用して Mobile Link サーバに接続できるようになりました。これによって、デバイス・トラッキングと通知でセキュリティを適用できます。

  「Listener 構文」『Mobile Link - サーバ起動同期』の -x の説明を参照してください。

デバイス・サポートの増加

- **Treo 600 と 650 のサポート** Palm Listener は、Treo 600 と 650 をサポートするようになりました。

- **CE Phone Edition のサポート** Listener で、SMS で Windows Mobile 2003 Phone Edition がサポートされるようになりました。

動作の変更と廃止される機能

次に、バージョン 10.0.0 で導入された Mobile Link の変更を示します。

Mobile Link サーバの変更

Mobile Link スクリプトの変更

- **カーソルベースのアップロードの削除** upload_cursor、new_row_cursor、old_row_cursor の各スクリプトは、バージョン 9.0.0 で廃止されました。代わりに、文ベースのスクリプトを使用してください。

  「ローをアップロードするスクリプトの作成」『Mobile Link - サーバ管理』を参照してください。

- **認識されないスクリプトにより同期が失敗する** Mobile Link サーバで認識されないテーブルレベルまたは接続レベルのスクリプトが検出されると、同期がアボートされます。以前のバージョンでは、認識されないスクリプトは警告メッセージを発行するだけでした。つまり、カーソルベースのアップロード・スクリプトが存在すると、同期がアボートします。

- **同期の失敗を引き起こすアップロード・スクリプトまたはダウンロード・スクリプトのエラー** Mobile Link サーバでアップロード・スクリプトやダウンロード・スクリプトでエラー

が検出されると、同期が常にアボートされるようになりました。以前は、同期がアボートされない場合がありました。

- **より制限されるようになった handle_error と handle_odbc_error イベントの動作**　handle_error と handle_odbc_error スクリプトが呼び出されるのは、アップロード・トランザクション中に Mobile Link で挿入、更新、または削除スクリプトの処理中に ODBC エラーが発生した場合、またはダウンロード・ローをフェッチしている場合だけになりました。それ以外の場合に ODBC エラーが発生すると、Mobile Link は report_error または report_odbc_error スクリプトを呼び出して、同期をアボートします。

- **認証スクリプトのコミット**　エラーがない場合、Mobile Link サーバは authenticate_user、authenticate_user_hashed、または authenticate_parameters を呼び出した後で常にトランザクションをコミットします。以前は、失敗した認証に関係するトランザクションがロール・バックされていたため、認証の試行失敗のレコードはありませんでした。次の項を参照してください。

  - 「authenticate_user 接続イベント」『Mobile Link - サーバ管理』
  - 「authenticate_user_hashed 接続イベント」『Mobile Link - サーバ管理』
  - 「authenticate_parameters 接続イベント」『Mobile Link - サーバ管理』

- **authenticate_user_hashed script の変更**　authenticate_user_hashed スクリプトは、ユーザの認証シーケンス中に複数回呼び出すことができるようになりました。

  「authenticate_user_hashed 接続イベント」『Mobile Link - サーバ管理』を参照してください。

- **開始スクリプトが呼び出されると、同期の成功に関係なく終了スクリプトが呼び出される**　Mobile Link スクリプトによっては、begin_connection と end_connection のように開始と終了の形式があります。以前は、同期に失敗すると、終了スクリプトが実行されないことがありました。今回のバージョンでは、開始スクリプトが呼び出されると、同期がエラーした場合でも、終了スクリプトが定義されていれば常に呼び出されるようになりました。

  「同期イベント」『Mobile Link - サーバ管理』を参照してください。

- **テーブルにアップロードするデータがない場合はアップロード・スクリプトが呼び出されない**　以前のバージョンでは、-us オプションを使用して、アップロードするデータがない場合は Mobile Link サーバがアップロード・スクリプトを呼び出さないようにすることができました。-us オプションは削除され、デフォルトではアップロード・ストリームにアップロードするデータが含まれる場合にかぎり、Mobile Link サーバはアップロード・スクリプトを呼び出します。-zus オプションを使用すると、以前の動作に戻すことができます。

  「-zus オプション」『Mobile Link - サーバ管理』を参照してください。

- **SQL Anywhere 10 と Microsoft SQL Server 2005 の統合データベースは、begin_connection スクリプトで独立性レベルを変更しない**　SQL Anywhere バージョン 10 と Microsoft SQL Server 2005 以上では、ダウンロードのデフォルトの独立性レベルがスナップショットになりました。つまり、ダウンロード・トランザクションの開始時に独立性レベルを変更できますが、その場合は begin_connection スクリプトの設定がすべて上書きされます。そのため、begin_download スクリプトのダウンロードの独立性レベルを変更するか、新しい mlsrv10 -dsd オプションを使用して、スナップショット・アイソレーションを無効にする必要があります。以前のマニュアルでは、begin_connection スクリプトで独立性レベルを変更することをおすすめしていましたが、スナップショット・アイソレーションを使用しない統合データベースに対しては、現在も有効な方法です。

「Mobile Link 独立性レベル」『Mobile Link - サーバ管理』を参照してください。

- **example_upload_cursor、example_upload_delete、example_upload_insert、example_upload_update テーブル・イベントの削除** -za と -ze の Mobile Link サーバ・オプションが削除されたことにより、example_upload_cursor、example_upload_delete、example_upload_insert、example_upload_update テーブル・イベントは生成されなくなりました。[同期モデル作成] ウィザードを使用してスクリプトを生成できるようになりました。

  「[同期モデル作成] ウィザード」『Mobile Link - クイック・スタート』を参照してください。

mlsrv10 の変更

- **-w と -wu オプションの変更** -w と -wu オプションは、データベース・ワーカ・スレッド数と、アップロードするデータベース・ワーカ・スレッドの最大数をそれぞれ設定します。以前のバージョンでは、これらのワーカ・スレッドが、ネットワークに対する読み込みと書き込み、プロトコル・バイトとロー・データのパックとアンパック、スクリプトの実行、統合データベースのローの更新とフェッチなど、同期のすべての処理を実行していました。

  今回のバージョンでは、-w と -wu で指定されたワーカ・スレッドはデータベース・ワーカ・スレッドになります。これらのデータベース・ワーカ・スレッドだけが、すべてのデータベース・アクティビティの処理のみを行います。他のスレッドは、ネットワーク・アクティビティ、パックとアンパック、その他の Mobile Link サーバのアクティビティの処理を行います。

  -w と -wu オプションの新しい動作は、ネットワークのアクティビティとは関係ありません。以前のバージョンでは、遅延時間が長いネットワークによってワーカ・スレッドがブロックされることがあり、一部の配備で大量のワーカ・スレッドを指定する必要がありました。新しい Mobile Link アーキテクチャでは、この要件はなくなりました。

  -w と -wu オプションは、Mobile Link サーバが統合データベースに与える負荷を制限する最も簡単な方法です。-w と -wu の値をテストすると、お使いの同期システムに最適なスループットを見つけることができます。次の項を参照してください。

  - 「-w オプション」『Mobile Link - サーバ管理』
  - 「-wu オプション」『Mobile Link - サーバ管理』
  - 「Mobile Link のパフォーマンス」『Mobile Link - サーバ管理』

- **キャッシュ・サイズの設定オプションの削除** 次の mlsrv10 オプションが削除されました。

  - -bc
  - -d
  - -dd
  - -u

  これらのオプションは、mlsrv10 -cm で置き換えられました。このオプションは、すべての同期でキャッシュを設定します。

  「-cm オプション」『Mobile Link - サーバ管理』を参照してください。

- **タイムアウトの設定オプションの削除** 次の mlsrv10 オプションは不要になり、削除されました。

  - contd_timeout

- unknown_timeout

また、mlsrv10 liveness_timeout オプションも削除されました。同期クライアント用のタイムアウト・オプションで置き換えられました。

「timeout」 『Mobile Link - クライアント管理』を参照してください。

- **必要なくなったバックログ・オプション** mlsrv10 バックログ・オプションは不要になり、削除されました。
- **プロトコル名とネットワーク・セキュリティ・オプションの変更** ネットワーク・プロトコル・キーワード https_fips、rsa_tls、rsa_tls_fips、ecc_tls と、ネットワーク・プロトコルのオプション・セキュリティが削除されました。プロトコルは削除されていませんが、異なる方法で指定する必要があります。mlsrv10 -x 構文は次のように変更されました。

以前の構文	バージョン 10.0.0 の新しい構文	説明
-x https_fips	-x https(fips=y;...)	HTTPS FIPS
-x rsa_tls	-x tls(tls_type=rsa;...)	RSA 暗号化を使用した TCP/IP TLS
-x rsa_tls_fips	-x tls (tls_type=rsa;fips=y;...)	RSA 暗号化と FIPS を使用した TCP/IP TLS
-x ecc_tls	-x tls(tls_type=ecc;...)	ECC 暗号化を使用した TCP/IP TLS
-x tcpip(security=...)	-x tcpip	TCP/IP
-x http(security=...)	-x http	HTTP

「-x オプション」 『Mobile Link - サーバ管理』を参照してください。

- **-bn オプションの変更** mlsrv10 -bn オプションは、競合検出時に BLOB バイトを比較します。以前は、LONGVARCHAR 型のデータの文字が比較されていました。今回のバージョンでは、バイナリと LONGVARCHAR BLOB で比較される単位は常にバイトになります。

「-bn オプション」 『Mobile Link - サーバ管理』を参照してください。

- **冗長出力の変更** mlsrv10 のオプション -vr、-vt、-vu は、どれもわずかに異なる情報を出力します。
  - **-vr** -vr は、アップロードとダウンロードのロー値だけを返すようになりました。以前は、アップロードとダウンロード・スクリプト名と内容も返していました。
  - **-vt** -vt は、変換後のスクリプトの内容だけを返すようになりました。以前は、元のスクリプトの内容も返していました。
  - **-vu** -vu は、未定義のテーブル・スクリプトを呼び出す必要があるときに、それらのスクリプトすべてを返すようになりました。これには統計スクリプトも含まれます。

「-v オプション」 『Mobile Link - サーバ管理』を参照してください。

- **Mobile Link サーバ・オプション -za と -ze の削除**　Mobile Link サーバの -za オプションと -ze オプションによる自動スクリプト生成は削除されました。[同期モデル作成] ウィザードを使用してスクリプトを生成できるようになりました。

  「[同期モデル作成] ウィザード」『Mobile Link - クイック・スタート』を参照してください。

- **-zac と -zec の削除**　Mobile Link サーバの、カーソルベースのスクリプトを生成する -zac と -zec オプションは廃止されていましたが、このバージョンで削除されました。

- **Mobile Link サーバ・オプション -oy の削除**　mlsrv10 -oy オプション (タイムスタンプの年を表す) が削除されました。情報、警告、エラーの各メッセージで、年はタイムスタンプに常に現れます。

統計プロパティ

- **統計プロパティの変更**　大きく 2 つの変更があります。

  - アップロードとダウンロードのバイト数の意味が変更されました。この数は、アップロードとダウンロードを格納するために Mobile Link サーバ内で使用されるメモリ量を反映するようになりました。以前は、Mobile Link サーバに対して送受信されたアップロードとダウンロードのバイト数を表していました。新しい数は、同期がサーバ・メモリに与える影響が反映されるので、より有用になりました。また、以前の数は、HTTP、暗号化、または圧縮が使用されたときの信頼性に欠けていました。

  - マニュアルの前のバージョンでは、プロパティを通常のアップロード・モードと強制的な競合モードのどちらで使用しているかによってプロパティが異なる値を返すことについて、プロパティの説明に記述されていませんでした。この点が修正されました (以下を参照)。

  次の統計プロパティが変更されました。

統計プロパティ	説明
conflicted_deletes	通常のアップロード・モードでは、この値は常に 0 です。 強制的な競合モードでは、upload_old_row_insert スクリプトで統合データベースに挿入された、アップロード済み削除の総数を返します。 以前は、アップロード済みで競合が検出された削除の数を返していました。
conflicted_inserts	通常のアップロード・モードでは、この値は常に 0 です。 強制的な競合モードでは、upload_new_row_insert スクリプトで統合データベースに挿入された、アップロード済み挿入の総数を返します。 以前は、アップロード済みで競合が検出された挿入の数を返していました。

統計プロパティ	説明
conflicted_updates	通常のアップロード・モードでは、競合が発生した更新ローの総数を返します。 強制的な競合モードでは、upload_new_row_insert スクリプトや upload_old_row_insert スクリプトで適用された、アップロード済み更新ローの総数を返します。 以前は、アップロード済みで競合が検出された更新の数を返していました。
download_bytes	ダウンロードを格納するために Mobile Link サーバ内で使用されるメモリ量を返します。 以前は、ダウンロード済みバイト数を返していました。
ignored_deletes	通常のアップロード・モードでは、upload_delete スクリプトが呼び出された場合、handle_error や handle_odbc_error が定義されていて 1000 が返された場合、または指定されたテーブルに定義された upload_delete スクリプトがない場合に、エラーが発生したアップロード済み削除ローの総数を返します。 強制的な競合モードでは、upload_old_row_insert スクリプトが呼び出された場合、handle_error や handle_odbc_error が定義されていて 1000 が返された場合、または指定されたテーブルに定義された upload_old_row_insert スクリプトがない場合に、エラーが発生したアップロード済み削除ローの総数を返します。 以前は、アップロード済みで無視された削除の数を返していました。
ignored_inserts	通常のアップロード・モードでは、upload_insert スクリプトが呼び出された場合、handle_error や handle_odbc_error が定義されていて 1000 が返された場合、または指定されたテーブルに定義された upload_insert スクリプトがない場合に、エラーが発生したアップロード済み挿入ローの総数を返します。 強制的な競合モードでは、upload_new_row_insert スクリプトが呼び出された場合、handle_error や handle_odbc_error が定義されていて 1000 が返された場合、または指定されたテーブルに定義された upload_insert スクリプトがない場合に、エラーが発生したアップロード済み挿入ローの総数を返します。 以前は、アップロード済みで無視された挿入の数を返していました。
ignored_updates	通常のアップロード・モードでは、upload_update スクリプトが呼び出された場合、handle_error や handle_odbc_error が定義されていて 1000 が返された場合、または指定されたテーブルに定義された upload_update スクリプトがない場合に、エラーが発生したアップロード済み更新ローの総数を返します。 強制的な競合モードでは、upload_new_row_insert または upload_old_row_insert スクリプトが呼び出された場合、または handle_error や handle_odbc_error が定義されていて 1000 が返された場合、エラーが発生したアップロード済み更新ローの総数を返します。 以前は、アップロード済みで無視された更新の数を返していました。

統計プロパティ	説明
upload_bytes	アップロードを格納するために Mobile Link サーバ内で使用されるメモリ量を返します。 以前は、アップロード済みバイト数を返していました。
upload_deleted_rows	通常のアップロード・モードでは、統合データベースから削除されたローの総数を返します。 強制的な競合モードでは、この値は常に 0 でした。 以前は、同期クライアントからアップロードされたロー削除の数を返していました。
upload_inserted_rows	通常のアップロード・モードでは、統合データベースで挿入されたローの総数を返します。 強制的な競合モードでは、この値は常に 0 です。 以前は、同期クライアントからアップロードされたロー挿入の数を返していました。
upload_updated_rows	通常のアップロード・モードでは、統合データベースで更新されたローの総数を返します。 強制的な競合モードでは、この値は常に 0 でした。 以前は、同期クライアントからアップロードされたロー更新の数を返していました。

「Mobile Link の統計のプロパティ」 『Mobile Link - サーバ管理』を参照してください。

Mobile Link サーバのその他の変更

- **リモート・データベースの CHAR または NCHAR データ型のカラムに対して Null 文字を同期できるようになった**　以前の Mobile Link では、null 文字を含む VARCHAR カラムと CHAR カラムの値によって同期が失敗することがありました。今回のバージョンでは、データ型 CHAR、VARCHAR、LONG VARCHAR、NCHAR、NVARCHAR、LONG NVARCHAR のリモート・データベース・カラムの null 文字を同期できるようになりました。
- **情報、警告、エラー・メッセージのログの新しいフォーマット**　以前は、Mobile Link サーバは次のフォーマットでメッセージをログに記録していました。

  T.*mm/dd hh:mm:ss. thread_id User_name: message*

  今回のバージョンでは、次のフォーマットでメッセージがログに記録されるようになりました。

  T. *yyyy-mm-dd hh:mm:ss. synchronization_id: message*

  同期ごとに、ログの最初のメッセージはリモート ID、ユーザ名、スクリプト・バージョン、クライアント名 (Ultra Light または SQL Anywhere) を示します。

  新しいフォーマットでは、提供される情報は減ることなく、出力ログのサイズが小さくなりました。

- **Oracle 用システム・プロシージャの新しいデータ型**　スクリプトを記録するための Mobile Link システム・プロシージャでは、スクリプトの内容パラメータで Oracle 統合データベース用の CLOB データ型を使用するようになりました。ml_add_property システム・プロシージャでは、prop_value パラメータが Oracle 用 CLOB になりました。以前は、これらのパラメータは VARCHAR 型でした。

  「Mobile Link サーバ・システム・プロシージャ」 『Mobile Link - サーバ管理』を参照してください。

Mobile Link クライアントの変更

Mobile Link ユーザ名とリモート ID

Mobile Link は、リモート ID と呼ばれるユニークな ID をリモート・データベースが初めて同期されるときに (または、NULL 値のリモート ID が出現したときに)、生成するようになりました。Mobile Link ユーザ名は、ユニークである必要がなくなりました。Mobile Link ユーザ名は、認証に使用される実際のユーザ名であると見なされるようになりました。

以前のバージョンでは、Mobile Link ユーザ名に対して同期の進行状況が格納されていました。今回のバージョンでは、SQL Anywhere リモートのリモート ID とサブスクリプション、Ultra Light リモートのリモート ID とパブリケーションに対して、進行状況が格納されるようになりました。

「ml_subscription」 『Mobile Link - サーバ管理』を参照してください。

以前は、Mobile Link ユーザ名を使用して、リモート・データベースをユニークに識別していました。Mobile Link ユーザに複数のリモート・データベースを同期させる場合に、リモート ID を使用してリモート・データベースを識別すると便利です。Ultra Light リモート・データベースでは、複数の Mobile Link ユーザが同じリモート・データベースを同期する場合に、リモート ID を使用すると便利です。

「リモート ID」 『Mobile Link - クライアント管理』を参照してください。

Ultra Light クライアント

「動作の変更と廃止される機能」 142 ページを参照してください。

SQLAnywhere クライアント

- **ダウンロード・エラー・フックの廃止**　エラー・フック sp_hook_dbmlsync_download_com_error、sp_hook_dbmlsync_fatal_sql_error、sp_hook_dbmlsync_sql_error は廃止されます。これらは置き換えられました。

  「イベント・フック・プロシージャ内でのエラーと警告の処理」 『Mobile Link - クライアント管理』を参照してください。

- **sp_hook_dbmlsync_log_rescan は dbmlsync が別の同期を予期した場合だけ呼び出される**　以前は、各同期の終了時に sp_hook_dbmlsync_log_rescan フックが呼び出されていました。そのため、dbmlsync が Mobile Link サーバから切断されてから、ログに同期の完了メッセージが表示されるまでに間がありました。今回のバージョンでは、dbmlsync が別の同期を予期した場合にかぎりフックが呼び出されるようになりました。たとえばコマンド・ラインで dbmlsync -n オプションを複数指定した場合や、スケジュールが有効な場合です。

「sp_hook_dbmlsync_log_rescan」『Mobile Link - クライアント管理』を参照してください。

- **活性タイムアウト・オプションの単純化** クライアントで、liveness_timeout と network_connect_timeout ネットワーク接続プロトコル・オプションが削除されました。代わりに timeout 接続オプションを使用してください。

  「timeout」『Mobile Link - クライアント管理』を参照してください。

- **圧縮をしない場合は難読化されない** 圧縮を none に設定した場合、データはまったく難読化されなくなりました。セキュリティ上問題がある場合は、トランスポート・レイヤ・セキュリティを使用して、データを暗号化してください。

  「compression」『Mobile Link - クライアント管理』を参照してください。

- **バージョン 7 の構文とユーティリティの削除** 次の SQL 文とユーティリティは廃止される予定でしたが、このバージョンで削除されました。
  - Mobile Link クライアント・データベース抽出ユーティリティ (mlxtract)
  - CREATE SYNCHRONIZATION SITE 文
  - CREATE SYNCHRONIZATION DEFINITION 文
  - CREATE SYNCHRONIZATION TEMPLATE 文
- **ActiveSync の新しいネットワーク・プロトコル・オプション** ActiveSync ユーザは、CommunicationAddress 拡張オプションまたは SQL 文で ADDRESS 句を指定するときに、ActiveSync プロトコルを指定する必要がなくなりました。代わりに、ActiveSync で、ActiveSync 用 Mobile Link プロバイダと Mobile Link サーバ間の通信に使用しているプロトコルとプロトコル・オプションを指定します。

  「Mobile Link クライアント・ネットワーク・プロトコル・オプション」『Mobile Link - クライアント管理』を参照してください。

- **dbmlsync の新しいシャットダウン方法** dbmlsync の -k オプションは廃止され、-qc オプションに置き換えられました。

  「-qc オプション」『Mobile Link - クライアント管理』を参照してください。

その他の Mobile Link の動作の変更

バージョン・サポート

- **バージョン 8.0.0 以前のクライアントのサポートの削除** Mobile Link サーバでは、バージョン 8.0.0 以前の SQL Anywhere クライアントがサポートされなくなりました。バージョン 10 の Mobile Link サーバで以前のデータベースを使用するには、次のアップグレード手順に従う必要があります。

  「SQL Anywhere のアップグレード」 350 ページを参照してください。

名前の変更

次のユーティリティ名が変更されました。

以前のユーティリティ名	新しいユーティリティ名
dbmlsrv9	mlsrv10
dbmluser	mluser
dbmlmon	mlmon
dbmlstop	mlstop
dbasinst	mlasinst

次のファイル名が変更されました。

以前のファイル名	新しいファイル名
dbmlsv9.dll	*mlodbc10.dll*
dbasdesk.dll	*mlasdesk.dll*
dbasdev.dll	*mlasdev.dll*
dbmlsrv.mle	*mlsrv10.mle*
syncasa.sql	*syncsa.sql*

ODBC ドライバの強化

Mobile Link では、Adaptive Server Enterprise と DB2 の統合データベース用に新しいドライバが使用されるようになりました。

「Mobile Link、QAnywhere、リモート・データ・アクセスで使用される ODBC の変更」 155 ページを参照してください。

サーバ起動同期

- **Windows SDK の削除**　多くの Windows デバイスのサポートを作成するための SDK が削除されました。向上した SMS サポートで置き換えられました。Palm SDK は残されます。
- **Listener -g オプションの置き換え**　dbsln -g オプションは dblsn -ni オプションで置き換えられました。

Mobile Link のその他の動作の変更

- **Windows パフォーマンス・モニタのサポート終了**　Mobile Link で、Windows パフォーマンス・モニタはサポートされなくなりました。代わりに Mobile Link モニタを使用してください。

  「Mobile Link モニタ」 『Mobile Link - サーバ管理』を参照してください。

- **サーバ起動同期でサポートされなくなった Kyocera デバイス**　Kyocera デバイス用の Palm SDK はなくなりました。Palm Listener では、引き続き Treo デバイスはサポートされます。

  「Palm 用 Mobile Link Listener SDK」 『Mobile Link - サーバ起動同期』を参照してください。

QAnywhere

次の項では、QAnywhere のバージョン 10.0.0 での新機能、動作の変更、廃止される機能について説明します。

新機能

次に、バージョン 10.0.0 で導入された QAnywhere の追加機能を示します。

モバイル Web サービス

モバイル Web サービスでは、モバイルに最適化された非同期 Web サービスのサポートを提供します。これによって、モバイル・アプリケーションでは、オフラインであっても Web サービス要求を行うことができ、あとで転送するためにそれらの要求をキューイングすることができます。要求は QAnywhere を使用してメッセージとして配信されます。サーバ側の Web サービス・コネクタは、クライアントの要求を Web サービスに転送します。次に、Web サービスから応答を受け取り、クライアントにメッセージとして返します。WSDL コンパイラが用意されているので、.NET または Java アプリケーションからモバイル Web サービスを使用できます。

「モバイル Web サービス」 『QAnywhere』を参照してください。

Sybase Central 用の新しい QAnywhere プラグイン

Sybase Central に QAnywhere プラグインが含まれるようになりました。このプラグインは、QAnywhere アプリケーションの作成と管理に使用する、使いやすいグラフィカルなインタフェースを提供します。QAnywhere プラグインでは、次の作業を実行できます。

- クライアント・メッセージ・ストアとサーバ・メッセージ・ストアの作成
- QAnywhere Agent の設定ファイルの作成と管理
- QAnywhere Agent のログ・ファイルのブラウズ
- 送信先エイリアスの作成または変更
- JMS コネクタと Web サービス・コネクタの作成
- 転送ルール・ファイルの作成と管理
- メッセージ・ストアのリモートでのブラウズ
- メッセージの追跡

QAnywhere は UNIX プラットフォームでサポートされていませんが、UNIX で Sybase Central を使用してメッセージを追跡できるようになりました。

新しい QAnywhere クライアント API

- **新しい SQL API** QAnywhere SQL API は、SQL ストアド・プロシージャのセットです。SQL 開発者は、簡単に QAnywhere メッセージング機能を使用できるようになります。この API を使用すると、既存のデータベース・アプリケーションを補完する簡単な方法で、ストアド・プ

ロシージャでメッセージを送受信できます。これにより、データベースとメッセージングの操作を 1 つのトランザクションに組み合わせる強力なアプリケーションが可能になります。たとえば単一のストアド・プロシージャで、ローをデータベースに挿入し、別のメッセージにメッセージを送信することができ、しかも両方のアクションを同じトランザクションの一部としてコミットできます。

「QAnywhere SQL API リファレンス」 『QAnywhere』を参照してください。

- **新しい Java クライアント API** Java 用の新しい QAnywhere クライアント API を使用すると、Java でメッセージング・クライアント・アプリケーションを作成できます。Java 用のクライアント API は、現時点では Windows CE を含む Windows のみでサポートされています。

  「QAnywhere Java API リファレンス」 『QAnywhere』を参照してください。

QAnywhere クライアント API の強化

QAnywhere クライアント API にインデックス処理に次のような追加が行われています。

- **メッセージ・セレクタ** SQL に似た式を使用して、キューからメッセージを選択的にブラウズしたり受け取ったりすることができるようになりました。メッセージ・セレクタを作成する構文は、転送ルールの条件に使用する構文と同一です。

  「QAnywhere メッセージの参照」 『QAnywhere』を参照してください。

- **メッセージをブラウズする新しい方法** 複数のキューからメッセージをブラウズしたり、ID やメッセージ・セレクタを基にメッセージのサブセットをブラウズしたりすることができるようになりました。

  「セレクタを使用したメッセージの参照」 『QAnywhere』を参照してください。

- **メッセージ・ストア・プロパティ名の列挙** メッセージ・ストア・プロパティ名を列挙できるようになりました。

  「クライアント・メッセージ・ストア・プロパティの列挙」 『QAnywhere』を参照してください。

- **配信不可メッセージ** 新しいメッセージ・ストア・プロパティ ias_MaxDeliveryAttempts を使用すると、QAnywhere クライアントが配信不可と判断する前にメッセージを受信しようとする最大試行回数を設定できます。

  「ルール変数」 『QAnywhere』を参照してください。

- **メッセージのキャンセル** メッセージを送信する前にキャンセルできるようになりました。

  「QAnywhere メッセージのキャンセル」 『QAnywhere』を参照してください。

- **メッセージ・ステータスのクエリ** 事前に定義された新しいメッセージ・プロパティ ias_Status と ias_StatusTime を使用して、メッセージのステータスを問い合わせられるようになりました。ias_Originator を使用してメッセージの発信者を問い合わせたり、ias_DeliveryCount を使用してメッセージが受信者に配信された回数を問い合わせたりすることができます。

  「事前に定義されたメッセージ・プロパティ」 『QAnywhere』を参照してください。

- **アップロードのインクリメントを設定する新しいメッセージ・ストア・プロパティ**　ias_MaxUploadSize を使用すると、アップロードのインクリメントを変更できます。

  「事前に定義されたクライアント・メッセージ・ストア・プロパティ」『QAnywhere』を参照してください。

QAnywhere Agent の新機能

- **単一デバイス上の複数エージェント**　以前は、1 つのデバイスで 1 つの QAnywhere Agent インスタンスしか実行できませんでした。この制限がなくなりました。

  「QAnywhere Agent の実行」『QAnywhere』を参照してください。

- **フェールオーバを設定するオプションの追加**　2 つの新しい QAnywhere Agent オプション -fd と -fr を使用すると、フェールオーバが発生する方法をカスタマイズできます。

  「-fd オプション」『QAnywhere』と「-fr オプション」『QAnywhere』を参照してください。

- **永続的な接続**　新しいオプション -pc+ が追加され、メッセージの転送用に永続的な接続を使用できるようになりました。新しい -push オプションは、-push_notifications を置き換えます。また、Push 通知で永続的な接続を使用するかどうかを指定することができるようになりました。

  次の項を参照してください。

  - 「-pc オプション」『QAnywhere』
  - 「-push オプション」『QAnywhere』

- **新しいアップグレード・プロシージャ**　新しい -sur オプションを使用すると、以前のバージョンの SQL Anywhere からのクライアント・メッセージ・ストアをアップグレードできます。

  「-sur オプション」『QAnywhere』を参照してください。

- **QAnywhere Agent によるメッセージのタイムスタンプへの ISO 8601 日時フォーマットの使用**　情報、警告、エラーの各メッセージのタイムスタンプは、明確に定義された ISO 8601 日時フォーマット (**{I|W|E}** *yyyy-mm-dd hh:mm:ss message*) を使用するようになりました。

その他の QAnywhere の強化

- **送信先エイリアス**　QAnywhere 送信先のセットを表す送信先エイリアスを定義できるようになりました。送信先エイリアスに送信されたメッセージは、エイリアスの各メンバに送信されます。

  「送信先エイリアス」『QAnywhere』を参照してください。

- **サーバ管理要求**　サーバ管理要求を使用して、アクティビティ (送信先エイリアスの作成や、JMS コネクタの監視、開始、停止など) を管理および監視できるようになりました。クライアントでサーバ管理要求を作成して、それらを処理するためにサーバ・メッセージ・ストアに送信できます。

  「サーバ管理要求」『QAnywhere』を参照してください。

- **サーバ側の転送ルールのメンテナンスの向上**　デフォルトのサーバ側の転送ルールを変更することができるようになりました。変更は、すべてのクライアントに自動的に適用されます。以前は、デフォルトを変更するには、各クライアントで転送ルールを手動で定義する必要がありました。

  「サーバ側の転送ルール」『QAnywhere』を参照してください。

- **さまざまなメッセージ・プロパティ**　事前に定義されたメッセージ・プロパティが QAnywhere に追加設定されました。メッセージの処理がより柔軟になり、デバッグ中の情報が詳細になり、メッセージのステータスをトラブルシューティングしやすくなりました。

  「メッセージ・プロパティ」『QAnywhere』を参照してください。

- **JMS 送信先にバックスラッシュを埋め込み可能**　バックスラッシュのデリミタが必要なサブコンテキストを JMS 送信先に含めることができるようになりました。

  「QAnywhere から JMS に送信されるメッセージのアドレス指定」『QAnywhere』を参照してください。

- **新しい転送ルール関数**　次の転送ルール関数が追加され、日付処理が向上しました。

  - DATEADD(datepart, count, datetime)
  - DATEPART(datepart, date)
  - DATETIME(string)
  - LENGTH(string)
  - SUBSTR(string, start, length)

  「ルール関数」『QAnywhere』を参照してください。

- **転送ルールのプロパティの前置**　メッセージ・プロパティやメッセージ・ストア・プロパティを転送ルールで使用するときに、これらの名前をプレフィクスとして使用して、同じ名前の転送ルール変数に指定された優先度をバイパスできるようになりました。

  「ルール変数としてのプロパティの使用」『QAnywhere』を参照してください。

動作の変更と廃止される機能

次に、バージョン 10.0.0 で導入された QAnywhere の変更を示します。

QAnywhere クライアントの変更

- **クライアント・メッセージ・ストア ID の変更**　クライアント・メッセージ・ストア ID は Mobile Link のリモート ID になりました。以前は、Mobile Link ユーザ名でした。統合データベースでリモート ID を登録する必要はありません。ただし、Mobile Link ユーザ名はサーバ・メッセージ・ストアで登録する必要があります。Mobile Link ユーザ名を指定しない場合は、デフォルトのクライアント・メッセージ・ストア ID になります。

  新しい qaagent オプションが、Mobile Link ユーザ名を管理するために提供されています。次の項を参照してください。

  - 「-mn オプション」『QAnywhere』

- 「-mp オプション」 『QAnywhere』
- 「-mu オプション」 『QAnywhere』

- **新しい ODBC ドライバ** Adaptive Server Enterprise と DB2 サーバ・メッセージ・ストアに接続するための iAnywhere Solutions ODBC ドライバは、変更されています。

  「Mobile Link、QAnywhere、リモート・データ・アクセスで使用される ODBC の変更」 155 ページを参照してください。

QAnywhere Agent の変更

- **qaagent -port の削除** -port オプションは、QAnywhere Agent が Listener からの通信を受信するポート番号を指定するオプションでした。このオプションは不要になり、削除されました。空きポートが自動的に使用されます。

- **qaagent -la_port の置き換え** -la_port オプションは -lp オプションで置き換えられました。

  「-lp オプション」 『QAnywhere』を参照してください。

- **qaagent -push_notifications の名前の変更** このオプションは -push という名前になりました。永続的な接続を使用した／使用しない Push 通知を使用できるようになりました。

  「-push オプション」 『QAnywhere』を参照してください。

- **ポリシーのデフォルトの変更** デフォルトのポリシーが自動 (automatic) になりました。以前のデフォルト値は、スケジュール済み (scheduled) です。デフォルトのスケジュール間隔は 900 秒 (15 分) になりました。以前のデフォルト値は、10 秒です。

  「-policy オプション」 『QAnywhere』を参照してください。

- **トランザクション・ログの使用と管理の終了** QAnywhere Agent では、トランザクション・ログが作成されなくなり、そのサイズも管理されなくなりました。その結果、ほとんどのアプリケーションでは、トランザクション・ログなしでデータベースを初期化する dbinit -n オプションを使用して、クライアント・メッセージ・ストアを作成する必要があります。

  「クライアント・メッセージ・ストアの設定」 『QAnywhere』を参照してください。

QAnywhere のその他の変更

- **サーバ側プロパティ・ファイルの廃止** プロパティをファイルに格納する代わりに、データベースに格納するようになりました。

- **getPropertyNames** getPropertyNames 関数が C++ クライアント API から削除されました。beginEnumPropertyNames、nextPropertyName、endEnumPropertyNames で置き換えられました。

  「QAMessage クラス」 『QAnywhere』を参照してください。

- **転送ルールでの日付処理** 次の転送ルール・メッセージ・ストア変数が削除されました。

  - ias_CurrentDayOfWeek
  - ias_CurrentDayOfMonth
  - ias_CurrentMonth
  - ias_CurrentYear

代わりに、ias_CurrentTimestamp または DATEPART を使用できます。

「ルール変数」『QAnywhere』を参照してください。

- **QAnywhere Central の置き換え**　QAnywhere Central は Sybase Central 用の QAnywhere プラグインで置き換えられました。このプラグインでは、機能に多くの強化が加えられています。

SQL Remote

次の項では、SQL Remote のバージョン 10.0.0 での新機能、動作の変更、廃止される機能について説明します。

新機能

次に、バージョン 10.0.0 で導入された SQL Remote の追加機能を示します。

- **invalid_extensions オプション**　新しいメッセージング・オプションが追加され、FILE と FTP メッセージングで特定のファイル拡張子を使用することにより、SQL Remote を停止できるようになりました。

  「SET REMOTE OPTION 文 [SQL Remote]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **UNIX と Linux プラットフォームで、Message Agent (dbremote) にグラフィカルなユーザ・インタフェースが備わりました。**　「Message Agent」『SQL Remote』の -ux オプションを参照してください。

- **dbremote によるメッセージのタイムスタンプへの ISO 8601 日時フォーマットの使用**　情報、警告、エラーの各メッセージのタイムスタンプは、明確に定義された ISO 8601 日時フォーマット ({**I**|**W**|**E**} *yyyy-mm-dd hh:mm:ss message*) を使用するようになりました。

- **dbremote の新しいオプション**　新しい -qc オプションを使用すると、完了時にウィンドウを閉じることができます。

  「Message Agent」『SQL Remote』を参照してください。

動作の変更と廃止される機能

次に、バージョン 10.0.0 で導入された SQL Remote の変更を示します。

- **Adaptive Server Enterprise データベースのサポートの停止**　SQL Remote では、Adaptive Server Enterprise 統合データベースがサポートされなくなりました。つまり、ssxtract、ssremote、ssqueue、その他すべての SQL Remote for Adaptive Server Enterprise ユーティリティとファイルがインストール環境から削除されました。

  Adaptive Server Enterprise データベースを同期するには、Mobile Link を使用する必要があります。

  SQL Remote から Mobile Link へのアップグレードについては、http://www.ianywhere.com/whitepapers/migrate_to_ml.html を参照してください。

- **[データベースの抽出] ウィザードで旧式のデータベースを抽出できなくなった**　[データベースの抽出] ウィザードは、バージョン 10 データベースだけで動作します。

- **#hook_dict で拡張された値**　ユーティリティ dbxtract と dbremote は、フックを公開し、テンポラリ・テーブル #hook_dict に名前／値ペアを値として渡します。以前は、#hook_dict テーブル内の値は VARCHAR (255) として定義されていました。これが VARCHAR (10240) に増加しました。
- **抽出ユーティリティの変更**　dbxtract には、いくつか変更が加えられています。
  - -j、-k、-x オプションは削除されました。
  - 新しいオプション -al と -xh が追加されました。

  「抽出ユーティリティ」『SQL Remote』を参照してください。
- **Message Agent (dbremote) の廃止されたオプション**　完了時にウィンドウを閉じるための -k オプションが廃止されました。新しい -qc オプションを使用すると、完了時にウィンドウを閉じることができます。

  「Message Agent」『SQL Remote』を参照してください。

Ultra Light

次の項では、Ultra Light のバージョン 10.0.0 での新機能、動作の変更、廃止される機能について説明します。

新機能

次に、バージョン 10.0.0 で導入された Ultra Light の追加機能を示します。

主な機能

Ultra Light は、管理のしやすさと SQL Anywhere の互換性を念頭に置いて設計された、完全な機能を備えた関係データベース管理システムになりました。新機能や便利な機能が追加されましたが、Ultra Light の専有容量は小さいままです。今回のリリースにおける Ultra Light の制限事項の詳細については、「データ・サイズとオブジェクトの制限事項」『Ultra Light - データベース管理とリファレンス』を参照してください。

今回のリリースに含まれる主な機能は次のとおりです。

- **データベースの上限の増加** Ultra Light データベースの上限が大幅に増加しました。実際に、テーブルのローの最大数は 1600 万にまで増加しました。その他の現在のデータベースの上限については、「データ・サイズとオブジェクトの制限事項」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **スキーマの統合** Ultra Light はスタンドアロンの RDBMS になり、データベースの論理構造を定義する個別のスキーマ・ファイルは必要なくなりました。今回のリリースでは、Ultra Light スキーマはデータベースに完全に統合されました。内部データベース・スキーマの詳細については、「Ultra Light データベース・スキーマ」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **統合されたファイル・フォーマット** バージョン 10 の Ultra Light では、ファイル・フォーマットが統合されました。これにより、ほとんどのプラットフォームでデータベース・ファイルを共有できるようになりました。必要な照合が定義されていない文字が必要な場合は、データベースのエンコードに UTF-8 を選択しなければなりません。詳細については、「Ultra Light での文字セットのエンコードに関するプラットフォーム要件」『Ultra Light - データベース管理とリファレンス』と「Ultra Light utf8_encoding プロパティ」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **データベース・パフォーマンスとデータ整合性の向上** 全体的に、Ultra Light のデータベース・パフォーマンスとデータ統合性は向上し、インデックス処理とデータベース・ページ管理は改善されました。
- **インデックスでハッシュ処理を利用可能** ハッシュ処理を利用するためにインデックスを指定できるようになりました。ハッシュ・サイズは、インデックス単位で指定できます。ハッシュ・サイズによってインデックス・ルックアップのパフォーマンスが向上でき、データベース・ファイル・サイズに影響することもあります。「Ultra Light クエリ・パフォーマンスの最適化 」『Ultra Light - データベース管理とリファレンス』を参照してください。

- **データベースの直接作成**　Ultra Light データベース・ファイルを直接作成できるようになりました。データベース・スキーマ・ファイルやリファレンス・データベース・ファイルは Ultra Light データベースのソースとして必要ありません。Sybase Central やコマンド・ライン・ユーティリティを使用したり、アプリケーションからプログラム的に実行して、Ultra Light データベースを個別に作成できます。

  既存の Ultra Light ユーザの場合は、以前のバージョンと同じ方法でデータベースを作成できなくなりました。「Ultra Light のアップグレード」 377 ページを参照してください。

- **Windows CE の直接サポート**　このリリースでは、Ultra Light アプリケーションはデスクトップから直接 Windows CE デバイスに配備されたデータベースに接続できます。Ultra Light データベースは、プレフィクス **WCE:¥** を付けてパスと名前を指定することにより、指定できます。これらの直接アクセスは、Sybase Central と Interactive SQL を含むすべてのクライアント・アプリケーションと管理ツールでサポートされています。「Windows CE」 『Ultra Light - データベース管理とリファレンス』を参照してください。

- **動的 SQL プログラミング・インタフェースとしての Embedded SQL**　以前のバージョンでは、Embedded SQL は静的なインタフェースでした。今回のバージョンでは、Ultra Light 動的 SQL のインタフェースとなり、SQL Anywhere データベースは不要になりました。Embedded SQL では、動的 ESQL 文や、ホスト変数のプレースホルダの使用もサポートしています。また、ESQL アプリケーションでは uleng10 を使用して実行できるようになりました。これは、*ulrt.lib* ではなく *ulrtc.lib* に対してリンクすることで実現できます。

  この変更の結果、簡単な Embedded SQL アプリケーションでもサイズが大きくなったり、複雑なアプリケーションでもサイズが小さくなったりする場合があります。「Ultra Light のアップグレード」 377 ページと「Embedded SQL アプリケーションの開発」 『Ultra Light - C/C++ プログラミング』を参照してください。

プラットフォームとデバイス

プラットフォームのサポートが変更されました。サポートされるプラットフォームのリストについては、「サポートされるプラットフォーム」 『SQL Anywhere 10 - 紹介』を参照してください。

特筆すべき重要な強化は次のとおりです。

- **配備プラットフォーム**　プラットフォームは次のように強化されています。
  - **Palm OS**　Palm OS デバイスの Ultra Light サポートが強化され、次のように変更されました。
    - Palm OS v4.x 以上のランタイム・サポートが提供されるようになりました。
    - CodeWarrior の開発サポートはバージョン 9 まで拡張されました。CodeWarrior 8 のサポートは終了しました。
    - 複数のデータベースと汎用のファイル名がサポートされるようになりました。DBF 接続パラメータを使用して、複数のデバイスに対して 1 つのデータベースを指定できるようになった。これにより、レコードベースとファイルベースのどちらの記憶領域を使用しているかによって、ファイル名が正しく設定されます。「Ultra Light DBF 接続パラメータ」 『Ultra Light - データベース管理とリファレンス』と「Ultra Light 接続パラメータ

でのファイル・パスの指定」『Ultra Light - データベース管理とリファレンス』を参照してください。

- NVFS デバイスと VFS デバイスがサポートされるようになりました。

- **Symbian OS** Symbian OS サポートは、今回のバージョンの Ultra Light で新しく追加されました。Ultra Light では、UIQ Phone (2.0 と 2.1)、Nokia S60 (second edition)、Series 80 の各デバイスにおける Symbian OS バージョン 7.0 と 8.0 をサポートします。

- **Windows Mobile 2005** Embedded Visual C++ 3.0 または 4.0 を使用している場合は、既存のランタイムを使用し続けることができます。ただし、Visual Studio 2005 を使用してアプリケーションを構築する場合は、新しいランタイム (*¥ultralite¥ce¥arm.50* にインストールされる) が必要です。

- **開発環境サポートの強化** 開発ツールと開発言語は次のように更新されました。
  - Ultra Light で、Visual Studio.NET 2003 での ADO.NET 1.0 開発と、Visual Studio 2005 での ADO.NET 2.0 開発がサポートされるようになりました。
  - Ultra Light では、Visual Basic と C# 開発用の AppForge Crossfire バージョン 5.6 がサポートされるようになりました。AppForge 用のアプリケーションは、Palm OS、Symbian OS、Windows CE の各プラットフォームに配備できます。
  - C++ コンポーネントの開発がサポートされるようになりました。

セキュリティ

- **暗号化タイプ** TLS で圧縮している場合、Ultra Light では ECC と RSA の両方の暗号化タイプがサポートされるようになりました。RSA 暗号化は個別の製品ではなくなりました。「トランスポート・レイヤ・セキュリティを使用する Ultra Light クライアントの設定」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **FIPS セキュリティ** FIPS セキュリティを使用して、Mobile Link サーバ通信をセキュリティ保護できるようになりました。

- **単純化されたセキュリティ・ストリーム** 個別のセキュリティ・パラメータを使用するのではなく、暗号化ストリームをネットワーク・プロトコルやストリーム・タイプとして定義できるようになりました。サポートされるストリーム・タイプは、TCP/IP、TLS (RSA、ECC、FIPS 用)、HTTP、HTTPS です。「Stream Type 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。

データベース管理

重要な新機能と強化は次のとおりです。

- **パスワードの変更** データベースで大文字と小文字を区別するかは関係なく、すべてのパスワードで大文字と小文字を区別します。新しいデータベースには、デフォルトのユーザ ID

DBA (パスワード **sql**) が作成されます。したがってユーザ ID、パスワード、信頼されるルート証明書は、以前のリリースからデータベースをアップグレードすると保存されません。

- **データベース・プロパティと接続パラメータの改善**　データベース・プロパティと接続パラメータが強化かつ単純化され、データベースと接続の動作を簡単に記述できるようになりました。今回のリリースのデータベースに設定できるデータベース・プロパティと接続パラメータの完全なリストについては、「Ultra Light データベース設定のリファレンス」『Ultra Light - データベース管理とリファレンス』と「Ultra Light の接続文字列パラメータのリファレンス」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **インデックスのパフォーマンスの向上**　今回のリリースで、Ultra Light のインデックスのパフォーマンスが強化されました。インデックスのハッシュ処理が導入されたことが大きな向上点の 1 つです。Ultra Light でインデックス・ハッシュ・サイズが設定できるようになりました。ハッシュ・サイズを 1 ー 32 バイトの値に設定することで、Ultra Light はインデックス・ページにインデックス付けされた値の一部またはすべてを格納します。これにより、必要なロー・ルックアップの回数が減少します。「Ultra Light でのインデックス・パフォーマンスの考慮事項」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **チェックサム検証**　データベース・ページがディスクに格納されるときにデータ整合性を検証するために、これらのページのチェックサムを含めることができるようになりました。「チェックサムを使用したページの整合性の確認」『Ultra Light - データベース管理とリファレンス』と「Ultra Light checksum_level プロパティ」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **BLOB サポートの拡張**　Ultra Light データベースで BLOB サポートが拡張されました。BLOB の更新、データ型のキャスト、長さの取得が可能です。

管理ツール

今回のリリースで管理ツールが強化されました。ツールを正しく使用できるように、マニュアルを確認してください。

グラフィカルな管理ツール

- **Sybase Central**　Sybase Central を使用して、Ultra Light データベースをグラフィカルなユーザ・インタフェースで作成、変更、管理できるようになりました。これは ulview スキーマ編集ユーティリティを置き換えるツールです。

  Sybase Central に含まれるウィザードのリストは次のとおりです。

  - [データベース作成] ウィザードを使用して、新しい Ultra Light データベースを構築します。このウィザードは ulcreate ユーティリティと同じ機能を共有します。
  - [データベース消去] ウィザードを使用して、既存の Ultra Light データベースを消去します。このウィザードと同じ機能のユーティリティはありません。
  - [データベースの抽出] ウィザードを使用して、SQL Anywhere リファレンス・データベースからの新しい Ultra Light データベースを初期化します。このウィザードは ulinit ユーティリティと同じ機能を共有します。

- [データベース・ロード] ウィザードを使用して、XML ファイルから Ultra Light データベースにデータをロードします。このウィザードは ulload ユーティリティと同じ機能を共有します。
- [C++ API 移行] ウィザードを使用して、ulgen ユーティリティ (削除されたユーティリティ) で作成された C/C++ コードをマイグレートします。このウィザードと同じ機能のユーティリティはありません。
- [データベース同期] ウィザードを使用して、Ultra Light データベースを同期します。このウィザードは ulsync ユーティリティと同じ機能を共有します。
- [データベース・アップグレード] ウィザードを使用して、以前のバージョンからの既存の Ultra Light データベースをアップグレードします。このウィザードは ulload ユーティリティと一緒に使用した場合の ulunloadold ユーティリティと同じ機能を共有します。
- [データベース・アンロード] ウィザードを使用して、データ／スキーマ情報を Ultra Light データベースから XML、SQL、または別のデータベースにアンロードします。このウィザードは ulcreate ユーティリティと ulload ユーティリティの追加機能と一緒に使用した場合の ulunload ユーティリティと同じ機能を共有します。

「Ultra Light プラグインのヘルプ」『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

- **Interactive SQL** Interactive SQL を使用して、Ultra Light データベースで SQL 文を開発し、テストできるようになりました。Interactive SQL は、以前のバージョンで使用された ulisql ユーティリティを置き換えます。「Ultra Light 用 Interactive SQL ユーティリティ (dbisql)」『Ultra Light - データベース管理とリファレンス』を参照してください。

コマンド・ライン管理ユーティリティ

次のコマンド・ライン・ユーティリティが Ultra Light に追加されました。

- **古いデータベースのアンロード・ユーティリティ** 新しい ulunloadold コマンド・ライン・ユーティリティを使用すると、既存の 8.0.2 または 9.x Ultra Light データベース (スキーマとデータ) またはスキーマ・ファイルを XML ファイルにアンロードできます。ulload コマンド・ライン・ユーティリティがあれば、Ultra Light バージョン 10 データベースを再構築するために、ユーティリティの出力を使用できます。「Ultra Light 古いデータベースのアンロード・ユーティリティ (ulunloadold)」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **情報ユーティリティ** 新しい ulinfo ユーティリティを使用すると、Ultra Light データベースに関する情報が表示されます。また、global_id や ml_remote_id などのデータベース・オプション ID を変更したりクリアしたりすることもできます。「Ultra Light 情報ユーティリティ (ulinfo)」『Ultra Light - データベース管理とリファレンス』を参照してください。

また、Ultra Light 10 の新しい RDBMS 機能をサポートするように既存のコマンド・ライン・ユーティリティが強化されたため、これらのユーティリティのオプションが以前のバージョンから変更されました。新しいユーティリティを正しく使用できるように、開始前にマニュアルを確認してください。詳細なユーティリティのリファレンス・ノートについては、「Ultra Light ユーティリティ・リファレンス」『Ultra Light - データベース管理とリファレンス』を参照してください。

- **エラー・レポート機能の強化**　Ultra Light のユーティリティは、他の SQL Anywhere ユーティリティと一貫性のあるエラーをレポートするようになりました。
- **データベース作成オプションの拡張**　すべてのデータベース作成ユーティリティ (ulcreate や ulload など) で、拡張された作成オプションを使用できるようになりました。これらの拡張オプションは、コマンド・ラインで -o を指定して設定でき、Sybase Central のウィザードで設定できるのと同じデータベース・プロパティを設定できます。拡張された作成オプションを適切に使用する方法の詳細については、「拡張作成時オプション」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **アンロード動作の強化**　ulunload を使用して、Ultra Light データベース・スキーマを動的 SQL 文のシーケンスとして出力できるようになりました。「Ultra Light データベースの XML へのアンロード・ユーティリティ (ulunload)」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **ulsync の動作の強化**　ulsync を使用すると、ネットワーク・プロトコル・オプションや拡張同期パラメータをこのユーティリティから直接設定できます。完全なリストについては、「Ultra Light 同期パラメータとネットワーク・プロトコル・オプション」『Mobile Link - クライアント管理』を参照してください。

  また、ulsync では、単なるパブリケーション・マスクではなく、パブリケーションに名前を付けることができるようになりました。キーワード **Publications** は、パブリケーション名のカンマ区切りのリストです。詳細については、「Ultra Light 同期ユーティリティ (ulsync)」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **コンジット・インストールの強化**　HotSync コンジット・インストール・ユーティリティ (ulcond10) で、コンジットの拡張機能、接続文字列、複数のデータベースがサポートされるようになりました。詳細については、「Palm OS 用 Ultra Light HotSync コンジットのインストール・ユーティリティ (ulcond10)」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **ulmvbreg**　AppForge 用 Ultra Light を登録するための ulmvbreg ユーティリティは、ulafreg に名前が変更されました。このユーティリティは *install-dir¥win32* ディレクトリにインストールされるようになりました。「Ultra Light AppForge レジストリ・ユーティリティ (ulafreg)」『Ultra Light - データベース管理とリファレンス』を参照してください。

ULSQLCONNECT

以前は、すべての Ultra Light ユーティリティがコマンド・ラインから接続情報を受け取っていました。今回のバージョンで、デフォルトのユーザ ID とパスワード以外の情報を渡す場合に、ホスト・マシンで ULSQLCONNECT 環境変数を設定できるようになりました。「ULSQLCONNECT 環境変数を使用した Ultra Light パラメータの保管」『Ultra Light - データベース管理とリファレンス』を参照してください。

SQL

- **SQL 文**　Ultra Light で、いくつかの新しい文がサポートされるようになりました。次のような新しい文があります。

- **ALTER TABLE**　Ultra Light SQL でテーブルを作成するほかに、この文で定義を変更できるようになりました。「Ultra Light ALTER TABLE 文」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **ALTER／CREATE／DROP PUBLICATION**　これら 3 つの文でパブリケーションの追加、作成、削除がサポートされるようになりました。「Ultra Light ALTER PUBLICATION 文」『Ultra Light - データベース管理とリファレンス』、「Ultra Light CREATE PUBLICATION 文」『Ultra Light - データベース管理とリファレンス』、「Ultra Light DROP PUBLICATION 文」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **START／STOP SYNCHRONIZATION DELETE**　Ultra Light SQL にこれらの文が含まれるようになりました。これらの文を使用して、Mobile Link 同期で削除がロギングされる方法を制御します。「Ultra Light START SYNCHRONIZATION DELETE 文」『Ultra Light - データベース管理とリファレンス』と「Ultra Light STOP SYNCHRONIZATION DELETE 文」『Ultra Light - データベース管理とリファレンス』を参照してください。

- **名前付き制約**　ALTER TABLE 文と CREATE TABLE 文で、テーブル制約に名前を付けることができるようになりました。このため、テーブル全体の制約を変更するのではなく、個々の制約を変更することで、テーブルやカラムの制約を変更できます。「Ultra Light ALTER TABLE 文」『Ultra Light - データベース管理とリファレンス』と「Ultra Light CREATE TABLE 文」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **その他の SELECT 文の強化**　SELECT 文が拡張されました。
  - SELECT 文の TOP 句に START AT を指定できるようになりました。START AT を指定すると、結果セットを明示的に制限するクエリの中で、さらに柔軟性を高めることができます。「Ultra Light SELECT 文」『Ultra Light - データベース管理とリファレンス』を参照してください。
  - DISTINCT 句が拡張され、この句で集合関数 (SUM、AVERAGE、MAX など) を使用できるようになりました。この句で集合関数を使用すると、実行時間が大幅に増加します。「Ultra Light SELECT 文」『Ultra Light - データベース管理とリファレンス』と「アルファベット順の関数リスト」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **UNION 演算子**　UNION 演算子を使用すると、2 つ以上のクエリから単一の結果セットを構築できます。デフォルトでは、UNION 演算子は、結果セットから重複しているローを削除します。「UNION 文を使用してセットを結合する」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

同期

- **設定可能で増加された HotSync コンジット同期のデフォルト・キャッシュ・サイズ**　以前は、Palm OS ファイルベースのデータ・ストアで同期されたデータが一定量を超えると、同期速度によくない影響がありました。これが、Ultra Light コンジット用のデフォルト・キャッシュ・サイズ (デスクトップ上) は 4 MB に増加されました。キャッシュ・サイズが大幅に増加したため、不要なファイル I/O 操作が減少し、同期時間が向上しました。ただし、選択した場合は異なるデフォルト・キャッシュ・サイズを設定することもできます。「Palm OS 用 Ultra Light HotSync コンジットのインストール・ユーティリティ (ulcond10)」『Ultra Light - データベース管理とリファレンス』を参照してください。

- **パブリケーションの述部**　Ultra Light 用の同期パブリケーションで述部が受け入れられるようになりました。条件式を論理演算子 AND や OR とオプションで組み合わせる場合、WHERE 句または HAVING 句に条件のセットを定義できるようになりました。SQL Anywhere と同様に、UNKNOWN と評価される述部が FALSE として解釈されます。「Ultra Light CREATE PUBLICATION 文」『Ultra Light - データベース管理とリファレンス』と「Ultra Light ALTER PUBLICATION 文」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **Mobile Link クライアント・ネットワーク・レイヤの向上**　次のクライアント・ネットワーク・レイヤが改善されました。
  - すべてのプロトコルで同期圧縮できます。
  - 永続的な接続によって、同じ接続で複数回同期できるようになりました。
  - IPv6 サポートが導入されました。
  - エラー検出とデバッグ機能が改善されました。

  Ultra Light を Mobile Link へのクライアントとして使用する場合の詳細については、「Ultra Light クライアント」『Mobile Link - クライアント管理』を参照してください。
- **同期用のテーブルの順序の設定**　Ultra Light クライアントからの同期に、テーブルのアップロード時に参照整合性の問題を避けるためにテーブルの順序を指定する機能が備わりました。同期のためにテーブルの順序を指定する場合は、table_order 同期パラメータを使用します。「Table Order 同期パラメータ」『Mobile Link - クライアント管理』または次の項目のいずれかを参照してください。
  - Ultra Light for MobileVB：「ULSyncParms クラス」『Ultra Light - AppForge プログラミング』
  - Ultra Light.NET：「ULSyncParms クラス」『Ultra Light - .NET プログラミング』
  - Ultra Light for C/C++：「ul_synch_info_a 構造体」『Ultra Light - C/C++ プログラミング』
  - Ultra Light for M-Business Anywhere：「SyncParms クラス」『Ultra Light - M-Business Anywhere プログラミング』
  - Ultra Light for Embedded SQL：「ULGetSynchResult 関数」『Ultra Light - C/C++ プログラミング』

プログラミング・インタフェース

一般的な向上点

- **カーソルの更新**　Ultra Light アプリケーションでは、カーソル処理中にデータベース内のデータを変更できる機能がサポートされるようになりました。SQL Anywhere データベースと同様に、すべてのクエリの結果セットで、カーソルの更新と削除ができるわけではありません。カーソルの更新が許可され、実行される場合について理解する必要があります。「データのフェッチ」『Ultra Light - C/C++ プログラミング』を参照してください。
- **単純化された接続文字列**　デフォルトのユーザ ID **DBA** とパスワード **sql** は常に Ultra Light によって提供されるため、接続文字列でデータベースを指定するだけで接続できるようになりました。さらに、ほとんどのデータベースを DBF 接続パラメータで設定できます。「Ultra Light データベースへの接続」『Ultra Light - データベース管理とリファレンス』を参照してください。

- **MLFileTransfer 関数の導入**　ファイル転送関数を使用すると、Mobile Link ファイル転送ユーティリティでファイルをダウンロードできます。ダウンロードされるファイルは、Mobile Link ユーザ名ごとに変えることも、デフォルトのファイルにすることもできます。たとえば、(月または処理サイクルの頭に) ローカル・データベースを置き換えるために、事前に設定された空のデータベース・ファイルをダウンロードするように選択できます。「Mobile Link ファイル転送ユーティリティ [mlfiletransfer]」『Mobile Link - クライアント管理』を参照してください。
  - Ultra Light for C++：「MLFileTransfer 関数」『Ultra Light - C/C++ プログラミング』
  - Ultra Light.NET：「ULFileTransfer クラス」『Ultra Light - .NET プログラミング』と「ULFileTransferProgressData クラス」『Ultra Light - .NET プログラミング』
  - Ultra Light for MobileVB：「ULFileTransfer クラス」『Ultra Light - AppForge プログラミング』
  - Ultra Light for M-Business Anywhere：なし
- **データベースの作成**　Ultra Light スキーマは、個別の *.usm* ファイルではなく、データベースの一部になりました。つまり、以前のバージョンでサポートされていたのと同じ方法では、アプリケーションが新しいデータベースを作成できなくなりました。

  次のいずれかを参照してください。
  - Ultra Light for C/C++：「ULCreateDatabase 関数」『Ultra Light - C/C++ プログラミング』
  - Ultra Light.NET：「ULDatabaseManager メンバ」『Ultra Light - .NET プログラミング』
  - Ultra Light for MobileVB：「ULDatabaseManager クラス」『Ultra Light - AppForge プログラミング』
  - Ultra Light for M-Business Anywhere：「 createDatabase メソッド」『Ultra Light - M-Business Anywhere プログラミング』

Ultra Light for C/C++

- **Symbian OS のサポート**　Symbian OS プラットフォーム用に、CodeWarrior または Carbide C++ 開発環境を使用した Ultra Light for C/C++ がサポートされるようになりました。
- **新しい関数**　今回のリリースには、さまざまな新しい関数が追加されています。これらの関数は次のとおりです。
  - GetPublicationMask 関数は、特定のパブリケーション名のパブリケーション・マスクを取得します。「IsCaseSensitive 関数」『Ultra Light - C/C++ プログラミング』を参照してください。
  - 特定のネットワーク・プロトコルで同期する前に、適切な ULEnable*Synchronization 関数を呼び出さなければならなくなりました。「ULEnableHttpSynchronization 関数」『Ultra Light - C/C++ プログラミング』、「ULEnableHttpsSynchronization 関数」『Ultra Light - C/C++ プログラミング』、「ULEnableTcpipSynchronization 関数」『Ultra Light - C/C++ プログラミング』、「ULEnableTlsSynchronization 関数」『Ultra Light - C/C++ プログラミング』、「ULEnableZlibSyncCompression 関数」『Ultra Light - C/C++ プログラミング』、または「ULEnableEccSyncEncryption 関数」『Ultra Light - C/C++ プログラミング』を参照してください。

- **ワイド文字とナロー文字 (ASCII) のサポートの向上**　Ultra Light のデータベース・ファイル・フォーマットは 1 種類 (ナロー文字) になりましたが、アプリケーションではワイド定義 TCHAR を使用し続けることができます。必要に応じて、ワイド文字とそれに等価な MBCS 文字の間で変換が行われます。

- **強化された関数の変更**　次のように、既存の関数が強化されました。

  - アプリケーションで SQL サポートが必要ない場合、ULInitDatabaseManager の代わりに ULInitDatabaseManagerNoSQL 関数を使用して、アプリケーションのサイズを大幅に小さくできます。「ULInitDatabaseManagerNoSQL 関数」『Ultra Light - C/C++ プログラミング』を参照してください。

  - SetReadPosition 関数が強化され、2 つ目のパラメータ offset_in_chars を取るようになりました。このパラメータは、オフセットがバイト単位か文字単位かを示します。「SetReadPosition 関数」『Ultra Light - C/C++ プログラミング』を参照してください。

  - SetSynchInfo で自動コミットが実行されるようになったため、すべての同期情報がすぐに保存されるようになりました。

  - ULStoreDefragInit、ULStoreDefragFini、ULStoreDefragStep は必要なくなりました。Ultra Light がデータベース・ストアの断片化解除を内部的に管理するようになりました。

  - ユーザ認証は常に有効なので、ULEnableUserAuthentication 関数は廃止されました。Ultra Light では、データベースに接続する可能性のあるユーザ名を最大 4 つまで定義できるようになりました (デフォルトのユーザ名は DBA、パスワードは sql)。

  - Mobile Link 同期コードで、InitSynchInfo の呼び出し前に ULEnableTcpipSynchronization を呼び出さなければならなくなりました。「InitSynchInfo 関数」『Ultra Light - C/C++ プログラミング』を参照してください。

Ultra Light Embedded SQL

Ultra Light Embedded SQL は静的な API ではなくなり、リファレンス・データベースが必要ではなくなりました。SQL プリプロセッサでは、ソース・ファイルだけが必要です。SQL プリプロセッサは、Ultra Light に SQL 文を送信する関数を生成します。以前のリリースでサポートされていたいくつかの SQL 文は、Ultra Light SQL でサポートされなくなりました。バージョン 10 では、以前のリリースでサポートされていなかった動的 SQL 文をサポートします。

- **新しい関数**　今回のリリースには、さまざまな新しい関数が追加されています。これらの関数は次のとおりです。

  - ULEnableZlibSyncCompression 関数で、同期時に zlib 圧縮が有効になりました。「ULEnableZlibSyncCompression 関数」『Ultra Light - C/C++ プログラミング』と「Mobile Link クライアントのネットワーク・プロトコル・オプション」『Mobile Link - クライアント管理』を参照してください。

    注意
    zlib 圧縮は Palm OS と Symbian OS でサポートされていません。

- **強化された関数**　次のように、既存の関数が強化されました。

- GetSQLColumnName が UltraLite_RowSchema_iface に追加されました。スキーマの種類によって、関数の返す値が異なります。
  - UltraLite_TableSchema で使用する場合、この関数は column_id パラメータで指定されたカラム名を返します。

    UltraLite_ResultSetSchema で使用する場合、この関数は次の値を返します。
    - 当該の結果セット・カラムでエイリアス名が指定されている場合は、エイリアス名
    - 結果セット・カラムがテーブルのカラムを表す場合は、カラム名
    - それ以外の場合は、すべて空の文字列

Ultra Light.NET

Ultra Light で、Visual Studio 2003 での ADO.NET 1.0 開発と、Visual Studio 2005 での ADO.NET 2.0 開発がサポートされるようになりました。

- **新しいメソッド**　今回のリリースには、さまざまな新しい関数が追加されています。これらの関数は次のとおりです。
  - ExecuteResultSet メソッドは SQL SELECT 文を実行して、更新可能な結果セットを ULResultSet クラスとして返します。「ExecuteResultSet メソッド」『Ultra Light - .NET プログラミング』を参照してください。
  - ULResultSet クラスには、Append*、Set*、Delete、Update の各メソッドが含まれます。これらのメソッドの詳細については、「ULResultSet クラス」『Ultra Light - .NET プログラミング』を参照してください。
  - Ultra Light.NET で、TCP/IP 同期時の TLS がサポートされるようになりました。「ULStreamType 列挙」『Ultra Light - .NET プログラミング』を参照してください。
  - ConnectionString プロパティと ULConnectionParms オブジェクトが強化され、限定的な引用符使用がサポートされるようになりました。「ULConnectionParms クラス」『Ultra Light - .NET プログラミング』を参照してください。
  - GetPublicationPredicate メソッドは、指定されたパブリケーションのパブリケーション述部文字列を返します。パブリケーションが存在しない場合は、SQLE_PUBLICATION_NOT_FOUND が設定されます。「GetPublicationPredicate メソッド」『Ultra Light - .NET プログラミング』を参照してください。
  - SignalSyncIsComplete メソッドは、ActiveSync 用の Mobile Link プロバイダに対して、アプリケーションが同期を完了したことを通知します。「SignalSyncIsComplete メソッド」『Ultra Light - .NET プログラミング』を参照してください。
  - SetDatabaseOption メソッドは、指定されたデータベース・オプションの値を設定します。「SetDatabaseOption メソッド」『Ultra Light - .NET プログラミング』を参照してください。
- **強化されたメソッド**　次のように、既存のメソッドが強化されました。

- ULSyncParms クラスに TableOrder 順序プロパティが用意されました。このプロパティは、統合データベースにテーブルがアップロードされる順序を指定します。「TableOrder プロパティ」『Ultra Light - .NET プログラミング』を参照してください。
- GetSchemaTable メソッドは、拡張 Table メタデータを返すようになりました。完全なリストについては、「GetSchemaTable メソッド」『Ultra Light - .NET プログラミング』を参照してください。
- テーブルが UL_TABLE_ACCESS_MODE_NONE または UL_TABLE_ACCESS_MODE_FIND_AGAIN の状態である場合、UpdateBegin メソッドは ResultSet レベルのオプションになりました。Ultra Light.NET API が ADO.NET 2.0 結果セットとの互換性を持たせるために、この変更が必要でした。「UpdateBegin メソッド」『Ultra Light - .NET プログラミング』を参照してください。
- GetDatabaseProperty メソッドは多くのプロパティを認識するようになりました。「GetDatabaseProperty メソッド」『Ultra Light - .NET プログラミング』を参照してください。
- ULSyncProgressData クラスに Flags プロパティが含まるようになりました。「Flags プロパティ」『Ultra Light - .NET プログラミング』を参照してください。

Ultra Light for AppForge Crossfire

Ultra Light for AppForge で Symbian OS プラットフォームがサポートされるようになりました。今回のリリースでは、Ultra Light エンジンのサポートが追加され、複数のアプリケーションが同時に単一データベースにアクセスできるようになりました。

- **新しいメソッド**　OnWaiting メソッドは、ユーザ・アプリケーションが GUI イベントを処理し、必要に応じて現在の操作をキャンセルするようなメカニズムを提供します。

Ultra Light for M-Business Anywhere

- **新しいメソッド**　今回のリリースには、さまざまな新しいメソッドが追加されています。これらの関数は次のとおりです。
  - setMBAServerWithMoreParms メソッドは、ワンタッチ同期を使用するときにプロキシ・サーバ情報を設定します。この新しいメソッドは、新しい文字列引数 **additional** が追加されたことで、既存の setMBAServer メソッドを強化します。
  - getPublicationMask メソッドは、特定のパブリケーション名のパブリケーション・マスクを取得します。「getPublicationMask メソッド」『Ultra Light - M-Business Anywhere プログラミング』を参照してください。
  - getPublicationPredicate メソッドは、指定されたパブリケーションのパブリケーション述部文字列を返します。パブリケーションが存在しない場合は、SQLE_PUBLICATION_NOT_FOUND が設定されます。
- **強化されたメソッド**　次のように、既存のメソッドが強化されました。

- setStream メソッドで、TLS (トランスポート・レイヤ・セキュリティ) の ECC (楕円曲線暗号) がサポートされるようになりました。「SyncParms クラス」『Ultra Light - M-Business Anywhere プログラミング』を参照してください。

> **注意**
> ECC 暗号化は、すべてのプラットフォームで使用できるわけではありません。サポートされるプラットフォームのリストについては、「サポートされるプラットフォーム」『SQL Anywhere 10 - 紹介』を参照してください。

動作の変更と廃止される機能

次に、バージョン 10.0.0 で導入された Ultra Light の変更を示します。

廃止されたプラットフォーム

- 今回のリリースで、PocketPC 2000 OS のサポートは廃止されます。
- CodeWarrior 8 はサポートされなくなりました。代わりに CodeWarrior 9 を使用する必要があります。
- Windows CE MIPS プロセッサはサポートされなくなりました。

サポートされているプラットフォームのその他の変更については、「SQL Anywhere がサポートするプラットフォームおよびエンジニアリング・サポート状況」の「SQL Anywhere 用 Ultra Light 展開オプション」を参照してください。

削除されたコンポーネント、モジュール、ネームスペース

今回のリリースから、次のプログラミング・インタフェースが削除されました。

- **Ultra Light ActiveX**　サポートされる API を使用して、すべてのアプリケーションを書き直す必要があります。
- **静的型 Java API**　サポートされる API を使用して、すべてのアプリケーションを書き直す必要があります。
- **Native Ultra Light for Java**　サポートされる API を使用して、すべてのアプリケーションを書き直す必要があります。
- **静的型 C++ API と静的型 Embedded SQL**　C++ アプリケーションを作成する開発者は、動的 C++ インタフェースを使用してプログラミングする必要があります。以前のバージョンから静的型 C++ ライブラリで作成されたアプリケーションがある場合、Ultra Light 10 にはこの新しいライブラリへの移動作業を簡単にする、マイグレーション・ユーティリティが含まれます。「Ultra Light のアップグレード」 377 ページを参照してください。
- **iAnywhere.UltraLite ネームスペース**　Ultra Light.NET では、このネームスペースがサポートされなくなりました。代わりに iAnywhere.Data.UltraLite ネームスペースを使用して、アプリケーションを書き直す必要があります。

削除されたユーティリティ

- **スキーマ・ペインタ**　Ultra Light データベースの作成でスキーマ・ファイルが不要になったため、スキーマ・ペインタ・ツールは削除されました。
- **データベース変換ツール**　データベース変換ツール (ulconv ユーティリティ) はサポートされなくなりました。ulconv の機能を実行する場合は、ulcreate、ulload、ulsync、ulunload の各ユーティリティを使用してください。
- **ulxml ユーティリティ**　スキーマ・ファイルを XML に変換する ulxml ユーティリティはサポートされなくなりました。ulxml に似た機能を実行する場合は、ulload と ulunload を使用して、データベースを XML に変換してください。
- **ulisql**　ulisql ユーティリティは、サポートされなくなりました。代わりに Interactive SQL (dbisql) で Ultra Light がサポートされるようになりました。
- **ulgen**　ulgen ユーティリティは、サポートされなくなりました。このユーティリティを使用する Ultra Light 配備では、データベースと C／C++ アプリケーションを適切にアップグレードする必要があります。「Ultra Light のアップグレード」 377 ページを参照してください。

削除、廃止、変更された関数

- **Ultra Light for C/C++ API**　C／C++ API で変更された関数やマクロは次のとおりです。
  - *.usm* ファイルが存在しなくなったため、データベース・スキーマは接続または動的にアップグレードできなくなりました。Ultra Light の以前の機能に関連するすべてのクラスと関数が削除されました。
  - 今回のバージョンでは、ULEnablePalmRecordDB と ULEnableFileDB が削除されました。
  - すべての ULEnableXXXX 関数は、初期化された SQLCA で呼び出されなければならなくなりました。
  - マクロ UL_STORE_PARMS がリリース 10 で廃止されます。OpenConnection または CreateDatabase の呼び出し時に、接続または作成のオプションが適切なパラメータで指定されます。
  - 今回のリリースで、ULSecureCerticomTLSStream と ULSecureRSATLSStream が廃止されます。これらに代わり、ULEccTlsStream と ULRsaTlsStream を使用できます。
  - ul_synch_info の security フィールドと security_parms フィールドが削除されました。代わりに、ストリーム・フィールドに適切な文字列 tcpip、http、https、または tls を設定してください。また、セキュリティ・パラメータを他のストリーム・パラメータと組み合わせてください。TCPIP は常に基本となるトランスポート・メカニズムです。HTTP 上の TLS はサポートされなくなりました。代わりに、HTTPS 同期ストリームを使用できます。「Ultra Light 同期パラメータとネットワーク・プロトコル・オプション」 『Mobile Link - クライアント管理』を参照してください。
  - ULSocketStream、ULHTTPStream、ULHTTPSStream は、必要な文字列を適切に返すように変更されました。
  - ULActiveSyncStream は Ultra Light から削除されました。ActiveSync 経由の同期では、アプリケーションはその他の種類の同期と同じようにストリーム値を指定する必要があります。

- **Embedded SQL**　C／C++ API に対する Embedded SQL インタフェースの関数は次のように変更されました。
  - *.usm* ファイルが存在しなくなったため、データベース・スキーマは動的にアップグレードできなくなりました。Ultra Light の以前の機能に関連するすべてのクラスと関数が削除されました。
- **Ultra Light.NET API**　Ultra Light.NET の関数は次のように変更されました。
  - *.usm* ファイルが存在しなくなったため、データベース・スキーマは接続または動的にアップグレードできなくなりました。Ultra Light の以前の機能に関連するすべてのクラスとメソッドが削除されました。
  - ULConnectionParms クラスで ParmsUsed プロパティの名前が ToString に変更されました。
  - GetSQLColumnName の名前が GetColumnSQLName に変更されました。
  - ULStreamType のメンバ UNKNOWN と ACTIVE_SYNC はこの列挙から削除されました。デフォルトは ULStreamType.TCPIP になりました。
- **Ultra Light for MobileVB API**　MobileVB API のメソッドは次のように変更されました。
  - *.usm* ファイルが存在しなくなったため、データベース・スキーマは接続または動的にアップグレードできなくなりました。Ultra Light の以前の機能に関連するすべてのクラスとメソッドが削除されました。
- **Ultra Light for M-Business Anywhere API**　M-Business Anywhere API の関数は次のように変更されました。
  - *.usm* ファイルが存在しなくなったため、データベース・スキーマは接続または動的にアップグレードできなくなりました。Ultra Light の以前の機能に関連するすべてのクラスとメソッドが削除されました。

名前の変更

- **ULUtil**　Palm OS 用の ULUtil ユーティリティの名前は ULDBUtil に変更されました。
- **ulmvbreg**　ulmvbreg の名前は ulafreg に変更されました。このユーティリティは *install-dir¥win32* ディレクトリにインストールされるようになりました。「Ultra Light AppForge レジストリ・ユーティリティ (ulafreg)」『Ultra Light - データベース管理とリファレンス』を参照してください。

その他

- **ulcond.log**　バージョン 10 の Ultra Light HotSync コンジット・インストーラ (ulcond10) は、このログファイルにメッセージを書き込まなくなりました。更新された ulcond10 ユーティリティの使用法については、「Palm OS 用 Ultra Light HotSync コンジットのインストール・ユーティリティ (ulcond10)」『Ultra Light - データベース管理とリファレンス』を参照してください。

Sybase Central と Interactive SQL

次の項では、バージョン 10.0.0 の Sybase Central と Interactive SQL での新機能、動作の変更、廃止される機能について説明します。

新機能

次に、バージョン 10.0.0 で導入された Sybase Central と Interactive SQL の追加機能を示します。

Sybase Central

この項では、Sybase Central の新機能について説明します。Sybase Central プラグインに対する変更や追加機能は、次の項で説明します。

- 「SQL Anywhere プラグイン」 145 ページ
- 「Sybase Central の新しいプラグイン」 146 ページ

- **Sybase Central タスク・リスト** Sybase Central の左ウィンドウ枠で、データベースのツリー構造ではなく、タスクを表示できます。タスク・リストには、現在選択されているオブジェクトに関連した共通のタスクが表示されます。タスク・リストには、共通のタスク、ナビゲーション・オプション、マニュアルへのリンクがあります。「ウィンドウ枠」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **新しい接続メニュー** 以前のリリースでは、[F11] キーを押して [新しい接続] ダイアログを開き、接続するプラグインを選択できました。今回のリリースでは、[接続] メニューが用意され、使用するプラグインを選択できるようになりました。[新しい接続] ダイアログは削除されましたが、[F11] キーを押すと、設定に使用するプラグインを選択できる [接続] メニューが開きます。
- **接続プロファイルの強化** Sybase Central 接続プロファイルに説明を追加できるようになりました。接続プロファイルはインポートやエクスポートもできます。
- **プラグインの検索** [ビュー] ▶ [検索ウィンドウ枠] を選択することで、Sybase Central でプラグインとデータベースから指定されたテキストを含むオブジェクトを検索できるようになりました。
- **コンテキスト・ドロップダウン・リスト** 新しいコンテキスト・ドロップダウン・リストには、現在オブジェクト・ツリーで選択されているオブジェクトが表示されます。これにより、特に左ウィンドウ枠でオブジェクト・ツリーを開いていない場合に、プラグイン間を簡単にナビゲーションできます。

SQL Anywhere プラグイン

- **[デッドロック] データベース・タブ** Sybase Central で SQL Anywhere データベースに接続しているときに、[デッドロック] タブでデッドロックについての情報を表示できます。「Sybase Central からデッドロックの表示」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **ER (実体関連) 図**　Sybase Central では、データベースの ER (実体関連) 図が表示され、データベース内のテーブルとその外部キーの関係が示されます。「SQL Anywhere プラグインからの ER 図の表示」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **新しい [スケジュール作成ウィザード] ウィザードとその他の [イベント] フォルダの強化**　スケジュールされたイベントの場合は、イベントがトリガされる次のスケジュールされた時刻が [イベント] フォルダに表示されるようになりました。条件イベントの場合は、このフォルダにはシステム・イベントと、オプションでそのイベントがトリガされるトリガ条件が表示されます。スケジュールは新しい [スケジュール作成] ウィザードを使用して作成されます。新しいスケジュールされたイベントを作成するときは、[イベント] ウィザードでイベントを作成できますが、必要に応じて後から追加スケジュールをイベントに追加できます。「スケジュールの定義」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **メンテナンス・プラン**　データベースの検証とバックアップを自動的に行うスケジュールを設定でき、その出力ログを電子メールで自分に送信できます。「メンテナンス・プランの作成」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **新しい [データベース・リストア] ウィザード**　[データベース・リストア] ウィザードを使用して、アーカイブ・バックアップからデータベースをリストアできるようになりました。「アーカイブ・バックアップのリストア」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **Sybase Central と Interactive SQL のエディタの強化**　[オプション] ダイアログの [フォーマット] タブで、Sybase Central と Interactive SQL のエディタ・ウィンドウで使用するフォントを選択できます。

  データベース・オブジェクト名の入力補完オプションがエディタに追加されました。「テキスト補完の使用」『SQL Anywhere サーバ - データベース管理』を参照してください。

Sybase Central の新しいプラグイン

- **QAnywhere プラグイン**　QAnywhere プラグインは、QAnywhere アプリケーションの作成と管理に使用する、使いやすいグラフィカルなインタフェースを提供します。

  「Sybase Central 用の新しい QAnywhere プラグイン」 122 ページを参照してください。
- **Ultra Light プラグイン**　Ultra Light プラグインを使用して、Ultra Light データベースをグラフィカルなユーザ・インタフェースで作成、変更、管理できるようになりました。

  「グラフィカルな管理ツール」 133 ページを参照してください。
- **Mobile Link の [同期モデル作成] ウィザードとモデル・モード**　ウィザードを使用して同期モデルを作成したり、新しいモデル・モードを使用して Mobile Link プラグインのモデルを編集したりすることができるようになりました。Mobile Link サーバ起動同期も設定できます。Mobile Link プラグインのこれまでの機能も強化され、管理モードで保存されます。

  「Sybase Central の Mobile Link プラグインの強化」 99 ページを参照してください。

Interactive SQL

- **Interactive SQL による Ultra Light データベースへの接続** Interactive SQL を使用して、Ultra Light データベースで SQL 文を開発し、テストできるようになりました。ulisql ユーティリティは廃止されています。「グラフィカルな管理ツール」 133 ページを参照してください。
- **Interactive SQL とサード・パーティのソース制御システムの統合** Interactive SQL はサード・パーティのソース制御システムと統合でき、Interactive SQL 内からファイルに対する一般的なソース制御操作 (チェック・イン、チェック・アウト、古いバージョンとの比較など) を実行できます。「ソース制御の統合」 『SQL Anywhere サーバ - データベース管理』を参照してください。
- **新しい Interactive SQL オプション** isql_maximum_displayed_rows オプションを使用すると、Interactive SQL の結果セットに表示されるローの数を指定できます。また、isql_show_multiple_result_sets オプションは、Interactive SQL の [結果] ウィンドウ枠に複数の結果セットを表示できるかどうかを指定します。「isql_maximum_displayed_rows オプション [Interactive SQL]」 『SQL Anywhere サーバ - データベース管理』と「isql_show_multiple_result_sets [Interactive SQL]」 『SQL Anywhere サーバ - データベース管理』を参照してください。
- **テキスト補完** Interactive SQL に、入力補完オプションが備わり、テーブル、ビュー、カラム、ストアド・プロシージャ、システム関数の名前を補完できるようになりました。「テキスト補完の使用」 『SQL Anywhere サーバ - データベース管理』を参照してください。
- **Interactive SQL で DESCRIBE 文のサポート** DESCRIBE 文を使用すると、指定されたテーブルやプロシージャについて次の情報を取得できます。

  - テーブルで見つかったすべてのカラム
  - テーブルで見つかったすべてのインデックス
  - ストアド・プロシージャで使用されるすべてのパラメータ

  「DESCRIBE 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **Interactive SQL の @data オプションのサポート** Interactive SQL をコマンド・プロンプトから開始するときに *@data* オプションを指定すると、指定された環境変数や設定ファイルからオプションを読み込むことができます。「Interactive SQL ユーティリティ (dbisql)」 『SQL Anywhere サーバ - データベース管理』を参照してください。

SQL Anywhere コンソール・ユーティリティ

- **SQL Anywhere の @data オプションのサポート** SQL Anywhere コンソールをコマンド・プロンプトから開始するときに *@data* オプションを指定すると、指定された環境変数や設定ファイルからオプションを読み込むことができます。「SQL Anywhere コンソール・ユーティリティ (dbconsole)」 『SQL Anywhere サーバ - データベース管理』を参照してください。

動作の変更と廃止される機能

次に、バージョン 10.0.0 で導入された Sybase Central と Interactive SQL の変更を示します。

- **Interactive SQL で quoted_identifier オプションが On に設定しなくなった**　以前のバージョンのソフトウェアでは、Interactive SQL は quoted_identifier オプションを On に設定していました。このオプションについては、データベースの設定が使用されるようになりました (デフォルトではこのオプションは On)。「quoted_identifier オプション [互換性]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **isql_plan オプションは NONE パラメータをサポートしなくなった**　isql_plan オプションの NONE パラメータは、サポートされなくなりました。「isql_plan オプション [Interactive SQL]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **Interactive SQL は無制限の結果セットを返すことができる**　以前のバージョンのソフトウェアでは、複数の結果セットを返すクエリを実行すると、Interactive SQL に表示される結果セットは最大で 10 個でした。今回のバージョンでは、クエリで返されるすべての結果セットが表示されるようになりました。「isql_show_multiple_result_sets [Interactive SQL]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **Interactive SQL の -f オプションの動作の変更**　Interactive SQL を -f オプションを使用して起動しても、データベースへの接続は自動的には確立されません。以前は、接続が自動的に開かれていました。
- **Interactive SQL でグラフィカルなプランのアクセスと保存**
  - グラフィカルなプランを開いたり保存したりするための 2 つの新しいメニュー項目 [プランを開く] と [プランの保存] が Interactive SQL の [ファイル] メニューから使用できるようになりました (以前は、SQL 文を開いたり保存したりする場合と同じメニュー項目 [開く] と [保存] を使用していました)。
  - 以前は、グラフィカルなプランは *.xml* ファイル拡張子で保存されていました。今回のバージョンでは、拡張子 *.saplan* で保存されるようになりました。ただし、*.xml* ファイル拡張子は、この拡張子で保存されたグラフィカルなプランを表示できるように、引き続きサポートされます。
  - [オプション] ダイアログで、*.sql* ファイルと *.saplan* (グラフィカルなプラン) ファイルのデフォルトのエディタとして Interactive SQL を指定できるようになりました。
- **SET OPTION 文の PUBLIC キーワードの廃止**　SET OPTION 文を使用して Interactive SQL を設定するための PUBLIC キーワードは廃止されます。「Interactive SQL オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **EXIT 文で現在の Interactive SQL ウィンドウを閉じるようになった**　以前のリリースでは、EXIT 文を Interactive SQL から実行すると、すべての Interactive SQL ウィンドウが閉じられました。今回のバージョンで、EXIT 文が実行されたウィンドウだけが閉じるようになりました。「EXIT 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **SQLE_ENGINE_NOT_MULTIUSER の新しいエラー・コード**　SQLE_ENGINE_NOT_MULTIUSER を処理するようにプログラミングしていたアプリケー

ションでは、新しいエラー・コードの検査が必要になりました。以前は、アプリケーションがデータベースで書き込み操作を行おうしたときに、別のスレッドがアップロードを Mobile Link に送信していた場合、SQLE_ENGINE_NOT_MULTIUSER が返されていました。今回のバージョンでは、より正確な新しいエラー・コード SQLE_ULTRALITE_WRITE_ACCESS_DENIED が返されるようになりました。「ULSQLCode 列挙」 『Ultra Light - .NET プログラミング』を参照してください。

- **Sybase Central プラグインの [ユーティリティ] タブの削除** SQL Anywhere、Mobile Link、Ultra Light の各プラグインの [ユーティリティ] タブは [ツール] ボタンで置き換えられました。ユーティリティは Sybase Central の [ツール] メニューからもアクセスできます。

非推奨機能とサポートされなくなった機能

- **jConnect による Sybase Central、Interactive SQL、SQL Anywhere コンソールへの接続サポート終了** Sybase Central、Interactive SQL、SQL Anywhere コンソール ユーティリティ (dbconsole) で、jConnect を使用した SQL Anywhere データベースへの接続がサポートされなくなりました。iAnywhere JDBC ドライバを使用して、これらのアプリケーションからデータベースに接続することはできます。この変更に伴い、次の機能が削除されました。
  - Interactive SQL ユーティリティ (dbisql) の -jconnect オプションと -odbc オプションが削除されました。
  - SQL Anywhere コンソール・ユーティリティ (dbconsole) の -jconnect オプションと -odbc オプションが削除されました。
  - Interactive SQL、SQL Anywhere コンソール、SQL Anywhere プラグインと Mobile Link プラグイン (Sybase Central) に接続するための [接続] ダイアログで、jConnect と iAnywhere JDBC ドライバのどちらを使用するかを指定できなくなりました。すべての接続で iAnywhere JDBC ドライバが使用されます。
- **Ultra Light プランが Interactive SQL の [プラン] タブに表示される** 以前のリリースでは、Interactive SQL の [結果] ウィンドウ枠に [Ultra Light プラン] タブがあり、このタブに Ultra Light プランの最適化方式が表示されていました。[Ultra Light プラン] タブは削除され、Interactive SQL から Ultra Light データベースに接続している場合は、[プラン] タブに表示されるようになりました。
- **Sybase Central の SQL Anywhere プラグインでバージョン 7 データベースのサポート終了** バージョン 7 データベース・サーバと、バージョン 7 ソフトウェアで作成されたデータベースのサポートが SQL Anywhere プラグインから削除されました。バージョン 5、6、または 7 ソフトウェアで作成したデータベースがバージョン 8 以降のデータベース・サーバで実行されている場合、データベースをアンロードして再ロード・ファイル、新しいデータベース、既存のデータベースに再ロードするために、そのデータベースに接続することはできます。「バージョン 9 以前のデータベースをバージョン 10.0.0 用に再構築」 360 ページを参照してください。
- **isql_log オプションの廃止** Interactive SQL セッション中に実行された文をロギングするための isql_log オプションは廃止されます。代わりに、START LOGGING 文と STOP LOGGING 文を使用してください。「コマンドのロギング」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Sybase Central ファイル名の変更**　Sybase Central の場合に説明したファイル変更のほかに、次のファイルが追加されました。

新しい名前	以前の名前
asaplugin.jar	*saplugin.jar*

Sybase Central の新しいバージョンを反映するように、レジストリ・キーも次のように変更されました。

HKLM¥SOFTWARE¥Sybase¥Sybase Central¥5.0 (registry entries)

マニュアルの強化

既存の機能の説明が、次に示すさまざまな分野で強化されています。

- **SQL 文のコンテキスト別ヘルプ**　Interactive SQL では、SQL 文の名前を右クリックして、その文の参照トピックを開くことができるようになりました。
- **Mobile Link のクイック・スタート**　新しいユーザが Mobile Link を使用して分散アプリケーションを開発する方法を理解できるように、新しい入門マニュアルが追加されました。

  Mobile Link - クイック・スタート『Mobile Link - クイック・スタート』を参照してください。
- **『データベース管理ガイド』に SQL Anywhere 管理ツールについての新しい章**　Sybase Central、Interactive SQL、SQL Anywhere コンソール・ユーティリティの使用方法について焦点を当てた新しい章が追加されました。

  「SQL Anywhere 管理ツール」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **EBF 用に更新されたサポートされているプラットフォームの情報**　製品とともにインストールされ、インストールするビルドを参照する、サポートされているプラットフォームのページが追加されました。これらのページは、SQL Anywhere インストール環境の *docs¥en* サブディレクトリにインストールされます。これらのページは、EBF で更新されます。マニュアルでは、必要に応じてこれらのページへのリンクがあります。
- **SQL Anywhere のセキュリティ機能の説明を移動**　『*SQL Anywhere Studio セキュリティ・ガイド*』というマニュアルがなくなりました。SQL Anywhere のセキュリティ機能については、『*SQL Anywhere - データベース管理*』に記載されています。
- **ODBC ドライバに関するマニュアルの削除**　『*ODBC Drivers for MobiLink*』というマニュアルはなくなりました。

製品全体の機能

次の項では、SQL Anywhere バージョン 10.0.0 のすべてのコンポーネントに影響する新機能、動作の変更、廃止される機能について説明します。

新機能

次に、バージョン 10.0.0 で導入された製品全体の追加機能を示します。

- **製品名が SQL Anywhere 10 に変更** SQL Anywhere Studio は SQL Anywhere 10 に、Adaptive Server Anywhere は SQL Anywhere に、それぞれ名前が変更されました。これに伴って、ファイル、ディレクトリ、サービス、実行可能ファイルの多くも名前が変更されました (ほとんどは ASA から SA への変更を反映したものです)。これらの変更の詳細については、この章の関連する「動作の変更」トピックで説明しています。
- **DataWindow.NET の新しいインストール** DataWindow.NET は、Visual Studio を使用するデータベース開発者に役立つ、カスタム制御ツールです。SQL Anywhere インストール中にオプションのコンポーネントとして提供されます。マニュアル全文はインストール環境に保存されます。
- **RSA が SQL Anywhere に付属** RSA 暗号化を使用するためのライセンスを別に購入する必要がなくなりました。「トランスポート・レイヤ・セキュリティ」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **機能の統計の収集** SQL Anywhere 10 は、ソフトウェアを実行するコンピュータの情報 (オペレーティング・システム・データベース・サーバの起動オプション、使用中の SQL Anywhere 10 ソフトウェア・ビルドなど) や、使用中の SQL Anywhere 10 機能の情報を追跡します。致命的なエラーが発生すると、問題についての情報と、SQL Anywhere 機能の統計値を iAnywhere に送信するかどうかを確認するプロンプトが自動的に表示されます。テクニカル・サポートに問題を報告すると、この情報を問題の診断に使用できます。「SQL Anywhere のエラー・レポート」『SQL Anywhere サーバ - データベース管理』を参照してください。

  SQL Anywhere サポート・ユーティリティ (dbsupport) を使用している場合はいつでも、機能の統計値を送信できます。機能の統計値情報は、iAnywhere が製品の使われ方を理解し、ソフトウェアを向上させるために役立ちます。「SQL Anywhere サポート・ユーティリティ (dbsupport)」『SQL Anywhere サーバ - データベース管理』を参照してください。

  SADIAGDIR 環境変数は、クラッシュ・レポートと機能の統計を格納するディレクトリのロケーションを指定します。「SADIAGDIR 環境変数」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **エラー・レポート** Sybase Central、Interactive SQL、SQL Anywhere コンソール・ユーティリティで内部エラーが発生すると、エラーのログがハード・ドライブに書き込まれ、エラー・レポートを iAnywhere に送信するかどうかを選択するダイアログが表示されます。

  また、パーソナル・サーバ、ネットワーク・サーバ、Mobile Link サーバ、dbmlsync、Mobile Link Listener、QAnywhere Agent、SQL Remote、または Replication Agent のいずれかで致命的なエラーが発生すると、エラーのログがハード・ドライブに書き込まれ、エラー・レポートを iAnywhere に送信するかどうかを選択するダイアログが表示されます。SQL Anywhere サポー

ト・ユーティリティ (dbsupport) は、これらのエラー・レポートを自動的に送信するように設定できます。「SQL Anywhere サポート・ユーティリティ (dbsupport)」を参照してください。

エラー・レポートを送信しないように選択すると、ファイルはハード・ディスクの診断ディレクトリに残ります。SADIAGDIR 環境変数は、クラッシュ・レポートと機能の統計を格納するディレクトリのロケーションを指定します。「SADIAGDIR 環境変数」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **管理ツールで使用できる機能の制御**　管理ツールでユーザが使用できる機能を制御できるようになりました。ツールの *.jar* ファイルと同じディレクトリにある *OEM.ini* ファイルを使用します。
- **Windows Vista での SQL Anywhere 10 の使用**　Windows Vista では、次のような既知の問題があります。
  - サーバ実行プログラムがあるディレクトリの管理者権限または書き込みパーミッションがない場合は、サーバ・ライセンス取得ユーティリティ (*dblic.exe*) を使用したライセンス情報の更新は失敗します。
  - SQL Anywhere をサービスまたは権限のないアカウントとして実行した場合、共有メモリの使用には既知の問題があります。この問題は、調査中です。代わりに、TCP/IP を使用できます。

動作の変更

次に、バージョン 10.0.0 で導入された製品全体の変更を示します。

- **サンプル・ディレクトリのロケーションの変更**　SQL Anywhere 10 で用意されているサンプルは、SQL Anywhere 10 のインストール環境にはインストールされなくなりました。この変更に伴い、サンプル・データベースは次のロケーションにインストールされるようになりました。

サンプル・データベース	ロケーション
SQL Anywhere サンプル・データベース	*samples-dir¥demo.db*
Mobile Link CustDB サンプル統合データベース・アプリケーション	*samples-dir¥MobiLink¥CustDB¥*
Ultra Light CustDB サンプル・データベース	*samples-dir¥UltraLite¥CustDB¥*

サポートされているオペレーティング・システムごとの *samples-dir* のデフォルト・ロケーションのリストについては、「サンプル・ディレクトリ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **サポートされなくなったプラットフォーム**　次のプラットフォームは、サポートされなくなりました。
  - Windows 95
  - Windows 98
  - Windows Me

- Windows NT
- Compaq Tru64

サポートされているプラットフォームのその他の変更については、「サポートされるプラットフォーム」『SQL Anywhere 10 - 紹介』を参照してください。

- **サンプル・データベースの改訂と名前の変更** SQL Anywhere サンプル・データベースの名前が *demo.db* になりました。画面読み上げソフトウェアを使用するユーザの便宜を図り、テーブル、カラム、ビュー、インデックスなどのオブジェクトは、単語が省略されていない名前になりました。「SQL Anywhere サンプル・データベース」『SQL Anywhere 10 - 紹介』を参照してください。

- **その他のファイル名の変更** 製品ごとに個別に説明したファイル名の変更のほかに、次のファイル名が変更されました。

以前の名前	新しい名前
asa.cvf	*sqlany.cvf*
asaldap.ini	*saldap.ini*
asasrv.ini	*sasrv.ini*
asa_config.sh	*sa_config.sh*
install-dir/SYBSasa9/lib	*sqlanywhere10/lib32* または *sqlanywhere10/lib64*
install-dir/SYBSasa9/bin	*sqlanywhere10/bin32* または *sqlanywhere10/bin64*

新しい製品名を反映するために、一部のレジストリ・キーが変更されました。次のようになりました。

HKLM¥SYSTEM¥CurrentControlSet¥Services¥Eventlog¥Application¥SQLANY
HKLM¥SYSTEM¥CurrentControlSet¥Services¥Eventlog¥Application¥SQLANY 10.0
HKLM¥SYSTEM¥CurrentControlSet¥Services¥Eventlog¥Application¥SQLANY 10.0 Admin
HKLM¥SOFTWARE¥Sybase¥Sybase Central¥5.0 (registry entries)
HKLM¥SOFTWARE¥Sybase¥SQL Anywhere¥10.0 (registry entries)

製品ごとに個別に説明したファイルの変更のほかに、*ulbase.lib* ファイルが追加されました。

次のサービス・グループ名が変更されました。

説明	以前の名前	新しい名前
ネットワーク・サーバ	ASANYServer	SQLANYServer
パーソナル・サーバ	ASANYEngine	SQLANYEngine
Mobile Link 同期クライアント	ASANYMLSync	SQLANYMLSync

説明	以前の名前	新しい名前
Replication Agent	ASANYLTM	SQLANYLTM

Mobile Link、QAnywhere、リモート・データ・アクセスで使用される ODBC の変更

- **Sybase Adaptive Server Enterprise ドライバ**　SQL Anywhere には、Adaptive Server Enterprise 用の iAnywhere Solutions ODBC ドライバが含まれなくなりました。代わりに、Adaptive Server Enterprise ネイティブ・ドライバが Mobile Link で動作するようにテストされています。iAnywhere Solutions 9 - Adaptive Server Enterprise Wire Protocol ドライバはサポートされなくなりました。

  http://www.ianywhere.jp/developers/technotes/odbc_mobilink.html を参照してください。

- **IBM UDB DB2 ドライバ**　SQL Anywhere には、DB2 用の iAnywhere Solutions ODBC ドライバが含まれなくなりました。代わりに、IBM DB2 8.2 CLI ドライバが Mobile Link で動作するようにテストされています。このネイティブ DB2 ドライバでは、DB2 バージョン 8.1 と 8.2 がサポートされます。IBM DB2 7.2 ODBC ドライバと iAnywhere Solutions 9 - DB2 Wire Protocol ドライバはサポートされなくなりました。

  http://www.ianywhere.jp/developers/technotes/odbc_mobilink.html を参照してください。

- **Oracle ドライバ**　iAnywhere Solutions 10 - Oracle Wire Protocol ドライバは、別個のダウンロードとして入手できます。

  http://www.ianywhere.jp/developers/technotes/odbc_mobilink.html を参照してください。

第 3 章

バージョン 9.0.2 の新機能

目次

バージョン 9.0.2 の新機能

この項では、SQL Anywhere Studio バージョン 9.0.2 のコンポーネントに導入された新機能について説明します。

Adaptive Server Anywhere の新機能

この項では、Adaptive Server Anywhere バージョン 9.0.2 の新機能について説明します。新機能について主要なものからそうでないものまですべてを示し、各機能の詳細についての参照先も記述しています。

SQL の強化

- **ネイティブの UNIQUEIDENTIFIER データ型** UNIQUEIDENTIFIER データ型が、BINARY(16) に基づいて定義されるドメインではなく、ネイティブのデータ型になりました。この結果、Adaptive Server Anywhere では、必要に応じて型変換が自動的に行われるため、STRTOUUID 変換関数と UUIDTOSTR 変換関数を使用して UNIQUEIDENTIFIER 値を処理する必要はありません。

  このリリースより前に作成されたデータベースで UNIQUEIDENTIFIER データ型を使用するには、データベースをアンロードして再ロードすることによってそのデータベースのファイル・フォーマットをアップグレードする必要があります。

  「UNIQUEIDENTIFIER データ型」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **RESOLVE UPDATE トリガの CONFLICT 関数** 競合解決トリガの中で CONFLICT 関数を使用して、SQL Remote 統合データベースに対して実行した UPDATE において、競合の原因が特定のカラムであるかどうかを判断できるようになりました。

  「CONFLICT 関数 [その他]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **プロシージャ・プロファイリングの強化** プロファイリング情報が、sa_server_option ストアド・プロシージャを使用して、ユーザ単位と接続単位でフィルタできるようになりました。

  「sa_server_option を使用したプロファイリングの有効化」『SQL Anywhere サーバ - SQL の使用法』と「sa_server_option システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **作成または変更前にリモート・サーバのテストが可能** Sybase Central の [リモート・サーバ作成] ウィザードに [テスト接続] ボタンが追加され、リモート・サーバを作成する前に、リモート・サーバ定義に指定されている接続情報を使用して正しく接続できるかどうかをテストできるようになりました。

  Sybase Central の [リモート・サーバ] プロパティ・シートにも [テスト接続] ボタンが追加され、プロパティを変更したときにリモート・サーバに正しく接続できるかどうかをテストできます。

「Sybase Central を使用したリモート・サーバの作成」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **INPUT 文と OUTPUT 文に ESCAPES 句を指定できる** ESCAPES 句を使用すると、データベース・サーバが文字を特殊文字として認識し、解釈することを指定できます。

  「INPUT 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』と「OUTPUT 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **別の接続からメッセージを受信して WAITFOR がウェイクアップできる** WAITFOR 文は、MESSAGE 文を使用して別の接続からメッセージを受信するときに、ウェイクアップできるようになりました。

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

- **抽出テーブルがクエリ・プランに表示される** 抽出テーブルが、クエリ実行プランにノードとして表示されるようになりました。

- **ALTER DOMAIN 文** ALTER DOMAIN 文を使用して、ユーザ定義のドメインとデータ型の名前を変更できます。

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

- **プロシージャの NO RESULT SET 句** 外部環境が、ストアド・プロシージャが結果セットを返さないことを理解する必要がある場合、ストアド・プロシージャの NO RESULT SET の宣言を使用することができます。

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

- **インデックス作成時にカラム統計が更新される** CREATE INDEX 文には、インデックス付けされたカラムの、カラム統計が更新されるという副次的な効果があります。

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

プログラミング・インタフェースの強化

- **PHP モジュール** SQL Anywhere PHP モジュールを使用すると、PHP スクリプト言語から Adaptive Server Anywhere データベースにアクセスできます。

  「SQL Anywhere PHP API」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **Web サービス・クライアント** Adaptive Server Anywhere は、Web サービス・プロバイダとして動作できるだけでなく、Web サービス・クライアントとしても動作できるようになりました。このため、インターネット上で使用できる標準の Web サービスに加えて、Adaptive Server Anywhere Web サービスにアクセスするストアド・プロシージャとストアド関数を作成できます。

  「SQL Anywhere Web サービス」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **複数の Web サービス・フォーマットのサポート** DISH サービスで提供される WSDL ファイルのフォーマットと、SOAP 応答の一部として返されるデータ・ペイロードのフォーマットを、クライアント・アプリケーションに合わせて選択できるようになりました。Microsoft .NET

用の DNET、自動的にインタフェースを生成するクライアント用の CONCRETE、汎用的な XML フォーマットの中から選択できます。

「SOAP および DISH Web サービスの作成」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **odbc_describe_binary_as_varbinary オプション** このオプションを使用すると、すべての BINARY カラムと VARBINARY カラムを、BINARY としてまたは VARBINARY として、アプリケーションに対して記述するかどうかを選択できます。

  「odbc_describe_binary_as_varbinary [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しい prefetch オプション値** prefetch オプションに、Always という新しい値が追加されました。この値は、SENSITIVE カーソル・タイプと、プロキシ・テーブルを含むカーソルであっても、カーソルの結果がプリフェッチされることを意味します。

  「prefetch オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **db_locate_servers_ex 関数** この関数を使用すると、特定のホスト上のすべての Adaptive Server Anywhere データベース・サーバをリストする、dblocate -n オプションで表示される情報に、プログラムからアクセスできます。

  「db_locate_servers_ex 関数」『SQL Anywhere サーバ - プログラミング』を参照してください。

管理の強化

- **SNMP エージェント** SNMP (Simple Network Management Protocol) アプリケーションから、Adaptive Server Anywhere をモニタできるようになりました。

  「SQL Anywhere SNMP Extension Agent」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **デッドロックのレポート** log_deadlocks という新しいデータベース・オプションと、sa_report_deadlocks という新しいシステム・ストアド・プロシージャを使用して、デッドロックに関係する接続についての情報を取得できるようになりました。log_deadlocks オプションをオンにすると、データベース・サーバは、デッドロックに関する情報を内部バッファに記録します。sa_report_deadlocks を呼び出すことによって、この内部バッファからデッドロック情報を取得できます。

  「ブロックされているユーザの判別」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **新しい照合** 今回のリリースには、次の照合が追加されています。

  - **1252SWEFIN** スウェーデン語とフィンランド語をサポートします。スウェーデン語とフィンランド語のシステムで照合が指定されていない場合、データベース・サーバは、新規データベースのデフォルトの照合として 1252SWEFIN を選択します。

  - **1255HEB** ヘブライ語をサポートします。ヘブライ語の Windows システムで照合が指定されていない場合、データベース・サーバは、新規データベースのデフォルトの照合として 1255HEB を選択します。

- **1256ARA**　アラビア語をサポートします。アラビア語の Windows システムで照合が指定されていない場合、データベース・サーバは、新規データベースのデフォルトの照合として 1255HEB を選択します。
- **950ZHO_HK、950ZHO_TW**　中国語をサポートします。950ZHO_HK は、Windows 中国語 (繁体字) 文字セット cp950 に加えて、香港補足文字セット (HKSCS) をサポートします。950ZHO_TW は、Windows 中国語 (繁体字) 文字セット cp950 はサポートしますが、HKSCS はサポートしません。順序は、中国語 (繁体字) のバイト単位の順序に基づきます。これらの照合が旧式の 950TWN 照合より優先されます。
- **1252SPA**　スペイン語をサポートします。スペイン語の Windows システムで照合が指定されていない場合、データベース・サーバは、新規データベースのデフォルトの照合として 1252SPA を選択します。
- **874THAIBIN**　タイ語をサポートします。Windows システムと UNIX システムの両方で、タイ語の照合として、この照合の使用が推奨されます。

「サポートされている照合と代替照合」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **サービス (dbsvc) ユーティリティの新しいオプション**　サービス・ユーティリティ (dbsvc) では、次のオプションが新しくサポートされるようになりました。
  - **-cm オプション**　このオプションは、指定されたサービスを作成するときに使用されたコマンドを表示します。これは、サービスを配備したり、元の状態に復元したりするときに便利です。
  - **-sd オプション**　このオプションを使用すると、Windows サービス・マネージャに表示される、サービスの説明を指定できます。
  - **-sn オプション**　このオプションを使用すると、Windows サービス・マネージャに表示される、サービスの名前を指定できます。

「Windows 用サービス・ユーティリティ (dbsvc)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **データ・ソース (dbdsn) ユーティリティの新しいオプション**　データ・ソース・ユーティリティ (dbdsn) では、次のオプションが新しくサポートされるようになりました。
  - **-cm オプション**　このオプションは、指定されたデータ・ソースを作成するときに使用されたコマンドを表示します。これは、データ・ソースを配備したり、元の状態に復元したりするときに便利です。
  - **Driver 接続パラメータ**　Windows でデータ・ソース・ユーティリティ (dbdsn) を使用してデータベース・ソースを作成する場合、Driver 接続パラメータを使用して、ODBC データ・ソースのドライバを指定できます。UNIX の場合は、Driver 接続パラメータを指定しないと、ASANY9 環境変数の設定に基づく Adaptive Server Anywhere ODBC ドライバのフル・パスを使用して、Driver エントリが自動的に追加されます。

「データ・ソース・ユーティリティ (dbdsn)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **ディスクが満杯の場合のコールバック・サポート** -fc データベース・サーバ・オプションを使用すると、ファイル・システムが満杯の状態になったときに、ユーザへの通知を行い、おそらく適切に対処できるコールバック関数が含まれる DLL を指定することができます。

  「-fc サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **[データベース検証] ウィザードの強化** Sybase Central で [データベース検証] ウィザードを使用してデータベースを検証する場合、ウィザードには、検証処理全体の進行状況に加えて、検証中の現在のテーブルも表示されるようになりました。また、チェックサムが有効なデータベースの場合は、テーブルとチェックサムを同時に検証できます。

  「データベースの検証」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Sybase Central でのテーブル・データのアンロード** [データのアンロード] ダイアログを使用して、Sybase Central 内の複数のテーブルから、1 回の操作でデータをアンロードできるようになりました。

  「[データのアンロード] ダイアログの使用」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **sa_index_density と sa_index_levels への新しいカラムの追加** sa_index_density および sa_index_levels ストアド・プロシージャから返される結果セットに、TableId、IndexId、IndexType という 3 つの新しいカラムが追加されました。これらのストアド・プロシージャの動作を以前の動作に戻すには、ストアド・プロシージャをいったん削除し、以前のバージョンのソフトウェアの結果セットに含まれていたカラムを指定して、ストアド・プロシージャを再作成します。

  「sa_index_density システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』と「sa_index_levels システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **BACKUP 文と RESTORE DATABASE 文の HISTORY オプション** HISTORY オプションを使用すると、BACKUP 文と RESTORE DATABASE 文による処理を *backup.syb* ファイルに記録するかどうかを制御できます。

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

- **Windows ユーザ・グループを使用する統合化ログインのサポート** Windows NT/2000/XP 上の個々のユーザに対して統合化ログインを作成できることに加えて、Windows NT/2000/XP 上のユーザ・グループに対しても統合化ログイン・マッピングを作成できるようになりました。データベースをアップグレードしてからこの機能を使用することをおすすめします。

  「Windows ユーザ・グループ用の統合化ログインの作成」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **要求ログのサイズの管理** -zn データベース・サーバ・オプションを使用すると、保持する要求ログ・ファイル数を指定できます。

  「-zn サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **バックアップによってファイルの名前が変更されるときにトランザクション・ログの末尾の空きページが削除される**　トランザクション・ログ・ファイルは、パフォーマンスを向上するために固定サイズで増分します。バックアップの一環でトランザクション・ログの名前が変更される場合、ログの末尾にある空きページが削除され、ディスク領域が解放されます。

- **リモート・サーバ接続を明示的に閉じることが可能**　以前のリリースでは、Adaptive Server Anywhere からリモート・サーバへの接続は、ユーザと Adaptive Server Anywhere との接続が切断された場合のみ、切断されていました。ALTER SERVER 文の新しい CONNECTION CLOSE 句を使用して、Adaptive Server Anywhere とリモート・サーバとの接続を明示的に切断できるようになりました。

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

セキュリティの強化

- **初期化ファイルを dbfhide で難読化できる**　ファイル非表示ユーティリティ (dbfhide) を使用して、Adaptive Server Anywhere とそのユーティリティが使用する .ini ファイルを難読化できるようになりました。

  「ファイル非表示ユーティリティ (dbfhide)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **FIPS 承認セキュリティ**　Windows CE を除く、サポートされているすべての Windows プラットフォームで、Certicom の FIPS 140-2 承認ソフトウェアで保護された、安全な通信を使用できるようになりました。

  「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」『SQL Anywhere サーバ - データベース管理』を参照してください。

  サポートされている 32 ビット Windows プラットフォームでは、Certicom の FIPS 140-2 承認ソフトウェアを使用した、強力なデータベース暗号化も使用できます。

  「データベースの暗号化」『SQL Anywhere サーバ - データベース管理』を参照してください。

その他の機能強化

- **新しい接続プロパティ**　次の接続プロパティが追加されました。
  - ClientPort
  - LoginTime
  - ServerPort

  「接続レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **イベント・ビューア・メッセージの適切なフォーマット**　Adaptive Server Anywhere データベースを配備する場合は、イベント・ビューアのメッセージのフォーマットを制御するレジストリ・エントリを設定する必要があります。

  「データベース・サーバの配備」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **log_deadlocks オプション**　このオプションを使用すると、データベース・サーバが、デッドロックに関する情報を内部バッファに記録するかどうかを制御できます。このオプションを sa_report_deadlocks プロシージャで使用すると、デッドロックに関する情報を取得できます。

  「log_deadlocks オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **rollback_on_deadlock オプション**　このオプションを使用すると、デッドロックが発生したときに、トランザクションを自動的にロールバックするかどうかを制御できます。

  「rollback_on_deadlock [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **temp_space_limit_check オプション**　このオプションを使用すると、接続がテンポラリ・ファイル領域の割り当てを越えて要求したときの動作を制御できます。

  「temp_space_limit_check オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しいシステム・ストアド・プロシージャ**　各種の新しいシステム・ストアド・プロシージャが追加されました。

  - **sa_rowgenerator プロシージャ**　sa_rowgenerator システム・プロシージャは、RowGenerator テーブルの代替として提供されており、指定された開始値と終了値の間のローから成る結果セットを返します。

    このプロシージャを使用すると、範囲内の各値に対応するローから成る結果セットを生成したり、結果セット内の既知の行数に対してテスト・データを生成したりできます。

    「sa_rowgenerator システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  - **sa_send_udp ストアド・プロシージャ**　このプロシージャは、指定されたアドレスに UDP パケットを送信し、Mobile Link サーバ起動同期で使用することで、リスナ・ユーティリティ (*dblsn.exe*) をウェイクアップできます。

    「sa_send_udp システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  - **sa_verify_password ストアド・プロシージャ**　このプロシージャは、現在のユーザのパスワードを確認するために、sp_password ストアド・プロシージャによって使用されます。

    「sa_verify_password システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

    「sa_verify_password システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **Windows CE の最大キャッシュ・サイズ**　SQL Anywhere Studio の以前のリリースでは、Windows CE の最大キャッシュ・サイズは 32 MB でした。この制限が取り除かれて、キャッシュ・サイズは、デバイス上で使用可能なメモリ量によって制限されるようになりました。

- **UNIX 用の新しいデータベース・サーバ・オプション**　UNIX 用に次のデータベース・サーバ・オプションが追加されました。
  - **-uc**　UNIX 上でデータベース・サーバをコンソール・モードで起動します。

    「-uc サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
  - **-ui**　X Windows がサポートされている Linux と Solaris でデータベース・サーバを起動する場合に、[サーバ起動オプション] ダイアログとサーバ・メッセージ・ウィンドウの表示を試みます。使用可能な表示を見つけられない場合は、サーバはコンソール・モードで起動します。

    「-ui サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
  - **-ux**　X Windows がサポートされている Linux と Solaris でデータベース・サーバを起動する場合に、[サーバ起動オプション] ダイアログとサーバ・メッセージ・ウィンドウを表示します。

    「-ux サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

Mobile Link の新機能

次に、バージョン 9.0.2 で導入したソフトウェアに加えられた変更と追加を示します。

- **新しいリダイレクタ**
  - Apache 用の新しいネイティブなリダイレクタは、Windows、Solaris、Linux で使用できます。
  - M-Business Anywhere リダイレクタは、Windows、Solaris、Linux で使用できます。

    「M-Business Anywhere リダイレクタ」『Mobile Link - サーバ管理』を参照してください。
  - NSAPI リダイレクタは、Solaris でも使用できるようになりました。以前は、Windows でのみ使用できました。

    「Windows 上の Netscape/Sun Web サーバ用の NSAPI リダイレクタ」『Mobile Link - サーバ管理』を参照してください。
- **指定のホストを無視するようにプロトコルを設定可能**　新しい ignore オプションを使用すると、Mobile Link サーバに接続するときに、サーバによって無視されるホストを指定できます。

  「-x オプション」『Mobile Link - サーバ管理』の **ignore** を参照してください。
- **Mobile Link サーバがビジーな場合のクライアントの同期待機の回避**　サーバがビジーな場合に、クライアントが同期を待ち続けることを回避できるようになりました。

  「-x オプション」『Mobile Link - サーバ管理』の **backlog** の説明を参照してください。

- **統合データベースに格納されるバージョン**　SQL Anywhere Studio のバージョンとビルド番号が、Mobile Link システム・テーブル ml_property に格納されるようになりました。エントリは、component_name が **ML**、property_set_name が **server_info**、property_name が **release_version**、property_value が *version.build* の形式で、9.0.2.1234 などがその例です。

  Mobile Link システム・テーブルの詳細については、「ml_property」『Mobile Link - サーバ管理』を参照してください。

- **Mobile Link サーバがサポートする新しい uniqueidentifier データ型**　UNIQUEIDENTIFIER データ型が、BINARY(16) に基づいて定義されるドメインではなく、ネイティブのデータ型になりました。この結果、Mobile Link リモート・データベースでは、必要に応じて型変換が自動的に行われるため、文字列から UUID への変換関数と UUID から文字列への変換関数を使用して UNIQUEIDENTIFIER 値を処理する必要はありません。

  サポートされている統合データベースへのこのデータ型のマッピングについては、「リモート・データベースと統合データベース間での Mobile Link データ・マッピング」『Mobile Link - サーバ管理』を参照してください。

セキュリティの強化

- **FIPS 承認セキュリティ・ストリーム**　Windows デバイスで、Certicom の FIPS 140-2 承認ソフトウェアが適用された安全な通信を使用できるようになりました。

  「トランスポート・レイヤ・セキュリティを使用する Mobile Link サーバの起動」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **出力ログ内での接続オプションの表示**　接続文字列と接続オプションが、出力ログに表示されるようになりました。このとき、パスワードはアスタリスクで表示されます。

- **旧式のセキュリティ機能**　「Mobile Link の動作の変更」 178 ページを参照してください。

Mobile Link クライアントの強化

- **Ultra Light 用の新しい同期設定ツール**　Ultra Light Schema Painter が、Adaptive Server Anywhere 統合データベース用のデータベース・テーブルとトリガに加えて、Mobile Link 同期スクリプトも生成できるようになりました。

- **リモート・データベースの削除と再作成の簡単化**　Adaptive Server Anywhere クライアント・サブスクリプションの最初の同期が、いつでも動作するようになりました。

  詳細については、「進行オフセット」『Mobile Link - クライアント管理』を参照してください。

- **Mobile Link への接続が失敗すると新しい dbmlsync フックが呼び出される**　新しく追加された sp_hook_dbmlsync_connect_failed イベント・フックを使用すると、同期接続が失敗した場合のリカバリの方法をプログラムできます。

  「sp_hook_dbmlsync_ml_connect_failed」『Mobile Link - クライアント管理』を参照してください。

- **Mobile Link クライアントと HTTP インフラストラクチャの統合の向上**　プロキシ・サーバか Web サーバまたはその両方が、RFC 2617 Basic または Digest 認証を要求する場合に、HTTP を使用して同期できるようになりました。

次の項を参照してください。

- 「http_password」『Mobile Link - クライアント管理』
- 「http_userid」『Mobile Link - クライアント管理』
- 「http_proxy_password」『Mobile Link - クライアント管理』
- 「http_proxy_userid」『Mobile Link - クライアント管理』

また、2 つの新しいクライアント接続パラメータを使用して、カスタム HTTP ヘッダとカスタム cookie を指定できるようになりました。セッション cookie に対処するために、HTTP クライアントは、サーバ応答で受信するすべての Set-Cookie および Set-Cookie2 HTTP ヘッダを認識し、これらの cookie を、この後のすべての HTTP 要求とともに送り返します。cookie の名前が既存の cookie に一致する場合、クライアントは、古い値を新しい値に置換します。同期と同期の間で cookie は記憶されません。同期の最後に破棄されます。

詳細については、「custom_header」『Mobile Link - クライアント管理』と「set_cookie」『Mobile Link - クライアント管理』を参照してください。

- **接続エラーの検出の支援**　Mobile Link クライアントは、無効な接続パラメータが指定された場合に警告メッセージを出力するようになりました。
- **ミラー・ログのロケーション**　dbmlsync が、リモート・データベースとは異なるコンピュータで実行している場合、または、ミラー・ログが、ミラー・トランザクション・ログとは異なるディレクトリにある場合、この新しい拡張オプションを使用して古いミラー・ログのロケーションを指定すると、dbmlsync は、古いログ・ファイルを自動的に削除できます。

  詳細については、「MirrorLogDirectory (mld) 拡張オプション」『Mobile Link - クライアント管理』を参照してください。

サーバ起動同期の強化

- **接続起動同期機能の強化**　Windows Listener は、_BEST_IP_CHANGED_ に加えて、内部メッセージ _IP_CHANGED_ も生成するため、接続に変更があった場合に同期を起動できるようになりました。

  詳細については、「接続起動同期」『Mobile Link - サーバ起動同期』を参照してください。
- **Listener ポスト・アクションの強化**　Listener ポスト・アクションを指定する場合、オプションで Windows メッセージ ID を使用して、ウィンドウ・メッセージを指定できるようになりました。また、ウィンドウ・クラスの代わりにウィンドウ・タイトルをオプションで使用できるようになりました。メッセージやタイトルに、スペースや句読点などの英数字以外の文字が含まれる場合は、ウィンドウ・クラス名やメッセージを単一引用符で囲むこともできます。

  詳細については、「Listener 構文」『Mobile Link - サーバ起動同期』の **post** を参照してください。
- **新しいアクション変数**　各種の新しいアクション変数が追加されました。
  - $request_id
  - $best_ip
  - $best_adapter_name

- $best_adapter_mac
- $best_network_name

詳細については、「action 変数」 『Mobile Link - サーバ起動同期』を参照してください。

- **デバイス・サポートの増加** Palm Listener は、Kyocera 7135 Smartphone と Treo 600 Smartphone をサポートするようになりました。

  詳細については、「Palm デバイス用 Listener」 『Mobile Link - サーバ起動同期』を参照してください。

SQL Remote の新機能

次に、バージョン 9.0.2 で導入したソフトウェアに加えられた変更と追加を示します。

- **ミラー・ログのロケーション** dbremote が、リモート・データベースとは異なるコンピュータで実行している場合、または、ミラー・ログが、ミラー・トランザクション・ログとは異なるディレクトリにある場合、新しい -ml オプションを使用して古いミラー・ログのロケーションを指定すると、dbremote は、古いログ・ファイルを自動的に削除できます。

  「Message Agent」 『SQL Remote』を参照してください。

- **RESOLVE UPDATE トリガの CONFLICT 関数** 競合解決トリガの中で CONFLICT 関数を使用して、SQL Remote 統合データベースに対して実行した UPDATE において、競合の原因が特定のカラムであるかどうかを判断できるようになりました。

  「CONFLICT 関数 [その他]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

Ultra Light の新機能

次に、バージョン 9.0.2 で導入したソフトウェアに加えられた変更と追加を示します。

コンポーネントの新しい機能

- **Ultra Light.NET の ADO.NET インタフェース** Ultra Light.NET は、新しい iAnywhere.Data.UltraLite ネームスペースで、ADO.NET プログラミング・インタフェースをサポートするようになりました。ADO.NET は、Ultra Light に対して業界標準のインタフェースを提供し、大規模なアプリケーションの、Adaptive Server Anywhere への簡単な移行パスも提供します。

  以前の Ultra Light.NET インタフェース (iAnywhere.UltraLite ネームスペース) に代わって ADO.NET インタフェースを使用してください。以前のインタフェースは推奨されなくなりました。

  「チュートリアル：Ultra Light.NET アプリケーションの構築」 『Ultra Light - .NET プログラミング』と「Ultra Light .NET 1.0 API リファレンス」 『Ultra Light - .NET プログラミング』を参照してください。

- **Ultra Light for MobileVB の強化** Ultra Light for MobileVB は、AppForge Crossfire を使用して、Visual Basic .NET プログラミングをサポートするようになりました。

  Ultra Light - AppForge プログラミング 『Ultra Light - AppForge プログラミング』を参照してください。

- **Ultra Light for M-Business Anywhere の強化** Ultra Light for M-Business Anywhere には、次のような強化が行われています。

  - Ultra Light for M-Business Anywhere は、クライアント／サーバ Ultra Light エンジンをサポートするようになりました。アプリケーションは、DatabaseManager.runtimeType プロパティを使用して、エンジンまたはランタイム・ライブラリが使用されているかどうかを調べることができます。
  - Ultra Light for M-Business Anywhere アプリケーションは、1 回の操作でデータと Web コンテンツの両方を同期できるようになりました。

    詳細については、「ワンタッチ同期」 『Ultra Light - M-Business Anywhere プログラミング』を参照してください。
  - Mobile Link リダイレクタを使用して、1 つの M-Business Anywhere Server を通じて、データと Web コンテンツの両方を同期できるようになりました。ファイアウォールの外部からの同期の場合、これによって、アクセス可能であることが必要なポート数が少なくて済みます。

    詳細については、「M-Business Anywhere を使用したデータの同期」 『Ultra Light - M-Business Anywhere プログラミング』と 「M-Business Anywhere リダイレクタ」 『Mobile Link - サーバ管理』を参照してください。
  - Windows XP 上の M-Business Anywhere 5.5 が、プラットフォームとしてサポートされるようになりました。接続パラメータの databaseOnDesktop と schemaOnDesktop が、この環境をサポートします。
  - API に追加されたメソッドを使用すると、カラム名ではなく、カラム ID を使用して、データに関する情報を収集できます。

    詳細については、「 ResultSetSchema クラス」 『Ultra Light - M-Business Anywhere プログラミング』 と 「 TableSchema クラス」 『Ultra Light - M-Business Anywhere プログラミング』 を参照してください。

  Ultra Light - M-Business Anywhere プログラミング 『Ultra Light - M-Business Anywhere プログラミング』を参照してください。

- **Native Ultra Light for Java の強化** Native Ultra Light for Java には、次のような強化が行われています。

  - 名前だけでなくカラム ID によってカラム・スキーマ情報にアクセス可能。
  - 新しい SyncProgressData ErrorMessage プロパティと、同期エラーのレポートの向上。
  - PreparedStatement.[get]Plan の追加。

- ResultSet と ResultSetSchema が使用されている間、PreparedStatement は有効。

- **Ultra Light.NET コンポーネントの強化**　次の機能が Ultra Light.NET でサポートされます。これらの機能は、ADO.NET インタフェース (iAnywhere.Data.UltraLite ネームスペース) の一部として使用することをおすすめします。
  - 新しい ULCursorSchema.Name、ULResultSetSchema.Name 読み込み専用プロパティ。
  - 新しい ULSyncProgressData ErrorMessage プロパティと、同期エラーのレポートの向上。
  - ULCommand.Plan 読み込み専用プロパティ。

  「Ultra Light .NET 1.0 API リファレンス」『Ultra Light - .NET プログラミング』を参照してください。

- **Palm 開発者はバージョンに依存しないプレフィクス・ファイルを使用できる**　以前のリリースでは、Ultra Light プレフィクス・ファイルは、開発している Palm OS のバージョンに依存していました。どのバージョンの Palm OS でも *ulpalmos.h* を使用できるようになりました。

  詳細については、「Ultra Light plug-in for CodeWarrior の使用」『Ultra Light - C/C++ プログラミング』を参照してください。

- **Palm 開発者が拡張モードを使用できる**　CodeWarrior は、グローバル・データのメモリの使用を向上する「**拡張モード**」と呼ばれるコード生成モードをサポートします。Ultra Light ランタイム・ライブラリの拡張モード・バージョンを使用できるようになりました。

  「拡張モード・アプリケーションの構築」『Ultra Light - C/C++ プログラミング』を参照してください。

- **信用された証明書を永続的な記憶領域から取得できる**　ソフトウェアの以前のリリースでは、安全な同期の信用された証明書は、データベース・スキーマに埋め込まれていました。Windows および Windows CE プラットフォームでは、信用された証明書を外部に保存できるようになり、trusted_certificates オプションを使用してアクセスできます。

  「trusted_certificates」『Mobile Link - クライアント管理』を参照してください。

SQL とランタイムの強化

- **動的 SQL の強化**　Ultra Light の動的 SQL のサポートには、次のような強化が行われています。
  - **クエリ最適化の向上**　ソフトウェアの以前のバージョンでは、テーブルがアクセスされる順序は、クエリ内でのテーブルの出現順序でした。このバージョンでは、テーブルに効率よくアクセスできる順序になるようにクエリが最適化されます。データベース内で適切なインデックスが定義されているかぎり、オプティマイザはクエリの実行パフォーマンスの向上を支援します。
  - **クエリ・プランの表示**　クエリ・アクセス・プランには、わかりやすいように、インデックス番号ではなくインデックス名が含まれるようになりました。アクセス・プランは、たとえば、新しい Ultra Light Interactive SQL ユーティリティから表示できます。
  - **IF 式と CASE 式**　IF と CASE の条件式が追加されて、Ultra Light がサポートする式の範囲が拡張されました。

「IF 式」『Ultra Light - データベース管理とリファレンス』と「CASE 式」『Ultra Light - データベース管理とリファレンス』を参照してください。

- **テーブル名に所有者名を指定できる** Ultra Light テーブルは、所有者を持ちません。既存の SQL とプログラムで生成された SQL の便宜上、*owner.table-name* がサポートされるようになりました。Ultra Light はこれを受け取りますが、*owner* は無視します。

- **UNIQUEIDENTIFIER データ型の導入** UNIQUEIDENTIFIER データ型が、BINARY(16) に基づいて定義されるドメインではなく、ネイティブのデータ型になりました。この結果、Ultra Light では、必要に応じて型変換が自動的に行われるため、文字列から UUID への変換関数と UUID から文字列への変換関数を使用して UNIQUEIDENTIFIER 値を処理する必要はありません。

  「UNIQUEIDENTIFIER データ型」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **Ultra Light クエリ・プランの記述の強化** Ultra Light Interactive SQL で表示できる Ultra Light クエリ・プランの記述が強化されて理解しやすくなり、パフォーマンスの問題を適切に診断できるようになりました。

管理の強化

- **Ultra Light Interactive SQL ユーティリティ** Ultra Light データベースに対する SQL 文のテストと、Ultra Light データの変更を行う目的で、Ultra Light Interactive SQL ユーティリティが提供されています。このユーティリティはクエリ・プランも表示するので、パフォーマンスの問題を診断できます。

- **データベース管理のコマンド・ライン・ユーティリティ** 一連のコマンド・ライン・ユーティリティによって、Windows コンピュータ上の Ultra Light ファイルを対象とするデータベース管理タスクが簡単になりました。これらのユーティリティは、アプリケーション開発時に特に効果的です。

  新しい各ユーティリティは、ulconv ユーティリティが提供するタスクのサブセットを実行します。ソフトウェアの将来のバージョンでは、ulconv ユーティリティが、これらの新しい単一タスク・ユーティリティに置き換わる予定です。

  次の項を参照してください。

  - 「Ultra Light データベース作成ユーティリティ (ulcreate)」『Ultra Light - データベース管理とリファレンス』
  - 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」『Ultra Light - データベース管理とリファレンス』
  - 「Ultra Light 同期ユーティリティ (ulsync)」『Ultra Light - データベース管理とリファレンス』
  - 「Ultra Light データベースの XML へのアンロード・ユーティリティ (ulunload)」『Ultra Light - データベース管理とリファレンス』

同期の強化

- **Mobile Link クライアントと HTTP インフラストラクチャの統合の向上** 2 つの新しいクライアント接続パラメータを使用して、カスタム・ヘッダとカスタム cookie を指定できるようになりました。

「custom_header」『Mobile Link - クライアント管理』と「set_cookie」『Mobile Link - クライアント管理』を参照してください。

- **Schema Painter からの同期スクリプトの生成** Ultra Light Schema Painter は、Adaptive Server Anywhere 統合データベースの同期スクリプトを生成できるようになりました。この機能を使用すると、Ultra Light アプリケーションを同期アーキテクチャに簡単に拡張できます。
- **参照整合性違反の同期通知** 参照整合性違反をレポートする同期コールバック関数がサポートされるようになりました。現在は、参照整合性に違反するローが通知されずに削除されます。

QAnywhere の新機能

- **フェールオーバ・サーバ** QAnywhere エージェントは、Mobile Link サーバ接続プロトコル・オプションを 1 つだけでなく、リストで持つことができるようになりました。

  「フェールオーバ・メカニズムの設定」『QAnywhere』を参照してください。
- **エミュレータのサポート** Pocket PC 2002 と Pocket PC 2003 の QAnywhere クライアント・アプリケーションは、x86 エミュレータをサポートするようになりました。これらのエミュレータでは、QAnywhere Agent の "scheduled" ポリシーのみがサポートされます。
- **サーバ・メッセージ・ストアとしてサポートされる新しい RDBMS** サポートされているすべての Mobile Link 統合データベースを、サーバ・メッセージ・ストア (Adaptive Server Anywhere、Adaptive Server Enterprise、Microsoft SQL Server、Oracle、DB2) として、QAnywhere アプリケーションの中で使用できるようになりました。
- **.NET Compact Framework の QAnywhere .NET クライアント・ライブラリによるメッセージ・リスナのサポート** .NET Compact Framework の QAnywhere .NET クライアント・ライブラリが、メッセージ・リスナをサポートするようになりました。

転送ルールの強化

- **リモート・メッセージ・ストア・プロパティの同期** リモート・メッセージ・ストア・プロパティを設定すると、そのプロパティがサーバ・メッセージ・ストアに同期され、転送ルールで使用できます。
- **メッセージ・ストア・プロパティの強化** ias_Network プロパティに、詳細なネットワーク情報にアクセスできるフィールドが設定されました。

  詳細については、「クライアント・メッセージ・ストア・プロパティ」『QAnywhere』を参照してください。

  さらに、カスタマイズされたメッセージ・ストア・プロパティを作成できます。
- **メッセージ削除のルール** メッセージ・ストア内でのメッセージの持続性について、転送ルールを指定できるようになりました。クライアント側とサーバ側でメッセージを削除できます。

  「メッセージ転送ルール」『QAnywhere』を参照してください。

QAnywhere Agent の強化

- **接続文字列**　qaagent -c オプションで接続文字列を指定して、ローカル・メッセージ・ストアを起動できるようになりました。Adaptive Server Anywhere 接続文字列パラメータを使用できます。

  詳細については、「-c オプション」 『QAnywhere』を参照してください。

- **クワイエット・モード**　QAnywhere Agent は 2 種類のクワイエット・モードをサポートしており、これによって、一部の Windows CE デバイスで生じる問題を回避できます。

  詳細については、「-q オプション」 『QAnywhere』および「-qi オプション」 『QAnywhere』を参照してください。

- **QAstop ユーティリティ**　-qi オプションを指定して QAnywhere Agent をクワイエット・モードで起動した場合は、新しい qastop ユーティリティを使用して QAnywhere Agent を停止する必要があります。

  詳細については、「-qi オプション」 『QAnywhere』を参照してください。

- **冗長性の強化**　-o または -ot オプションで出力・ログ・ファイルの名前を指定し、-os および -ot オプションで出力ファイルのサイズを調整できるようになりました。また、以前の -verbose オプションが -v オプションに置き換えられました。-v を使用すると、ログ出力を詳細に制御できます。

  次の項を参照してください。

  - 「-o オプション」 『QAnywhere』
  - 「-ot オプション」 『QAnywhere』
  - 「-on オプション」 『QAnywhere』
  - 「-os オプション」 『QAnywhere』
  - 「-v オプション」 『QAnywhere』

- **リモート・メッセージ・ストアとして使用するデータベースの初期化**　新しい qaagent -si オプションを使用して、リモート・メッセージ・ストアを設定できるようになりました。「-si オプション」 『QAnywhere』を参照してください。

- **バージョン 9.0.1 からのアップグレード**　QAnywhere Agent の新しい -su オプションを使用すると、バージョン 9.0.1 のリモート・メッセージ・ストアを 9.0.2 にアップグレードできます。

  「-su オプション」 『QAnywhere』を参照してください。

QAnywhere Mobile Link システム・テーブル

すべての QAnywhere Mobile Link システム・テーブルは、ml_qa_user_group によって所有されるようになりました。以前は、DBO によって所有されていました。

2 つの新しい Mobile Link システム・テーブルが追加されました。次の項を参照してください。

- 「ml_qa_delivery」 『Mobile Link - サーバ管理』
- 「ml_qa_delivery_client」 『Mobile Link - サーバ管理』

各種の Mobile Link システム・テーブルのスキーマに変更が加えられています。次の項を参照してください。

- 「ml_qa_global_props」『Mobile Link - サーバ管理』
- 「ml_qa_global_props_client」『Mobile Link - サーバ管理』
- 「ml_qa_repository」『Mobile Link - サーバ管理』
- 「ml_qa_repository_client」『Mobile Link - サーバ管理』
- 「ml_qa_repository_props」『Mobile Link - サーバ管理』
- 「ml_qa_repository_client」『Mobile Link - サーバ管理』

9.0.2 クライアントの場合、次の Mobile Link システム・テーブルは生成されません。

- ml_qa_repository_staging_client
- ml_qa_status_staging_client

9.0.2 サーバの場合、次の Mobile Link システム・テーブルは生成されません。

- ml_qa_repository_content

マニュアルの強化

この項では、バージョン 9.0.2 の Adaptive Server Anywhere マニュアルの体裁、編成、またはナビゲーションの強化について説明し、主な変更についてはすべて説明します。

新しいマニュアル

既存の機能の説明が、次に示すさまざまな分野で強化されています。

- **SNMP エージェントのマニュアル** Adaptive Server Anywhere SNMP エージェントを説明する新しいマニュアルが追加されました。

  「SQL Anywhere SNMP Extension Agent」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Windows CE の手引き** Windows CE ユーザの手引きとなる章が追加されました。

  「SQL Anywhere for Windows CE」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Mobile Link 同期クライアントに対する DBTools インタフェース** DBTools の dbmlsync の使用方法に関するサンプルとその他の情報が追加されました。

  詳細については、「dbmlsync の DBTools インタフェース」『Mobile Link - クライアント管理』を参照してください。

- **QAnywhere の強化**　QAnywhere のマニュアルが拡張され、JMS メッセージング・システムと Mobile Link データ同期にメッセージングを統合する方法に関して、新しい情報が追加されました。また、QAnywhere アプリケーションの設定に関する情報が強化されました。

  QAnywhere 『QAnywhere』を参照してください。

- **サーバ起動同期の SDK**　SDK のマニュアルが拡張され、Palm Listener SDK の項が新しく追加されました。

  「Palm 用 Mobile Link Listener SDK」 『Mobile Link - サーバ起動同期』を参照してください。

マニュアルの強化

- **Mobile Link の再編成**　Mobile Link のマニュアルが、クライアント・ガイド、管理ガイド、チュートリアル・マニュアルに再編成されました。クライアント・ガイドには、Adaptive Server Anywhere クライアントの説明に加えて、以前は『Ultra Light データベース・ユーザーズ・ガイド』に記載されていた、Ultra Light クライアント用の同期パラメータと同期接続パラメータの説明も含まれています。

- **Ultra Light API と QAnywhere API のリファレンス**　Ultra Light.NET、Ultra Light C++ API、QAnywhere .NET、QAnywhere C++ API の情報が、他のマニュアルと同じ形式で提供されるようになりました。この結果、HTML ベースのマニュアルと同様、PDF でも提供されます。

バージョン 9.0.2 での動作の変更

この項では、SQL Anywhere Studio バージョン 9.0.2 のコンポーネントに導入された動作の変更について説明します。また、現在のソフトウェアではサポートされているが、SQL Anywhere Studio の今後のメジャー・リリースからはサポートされなくなる機能についても説明します。

廃止される機能は変更の可能性があります
廃止される機能のリストはあくまでも予定であって完全なものとは限らず、変更の可能性があります。

Adaptive Server Anywhere の動作の変更

非推奨機能とサポートされなくなった機能

ここでは、サポートを終了した機能、今後使用することを推奨しない機能の中で、既存のアプリケーションに影響する可能性があるものがリストされています。

- **min_table_size_for_histogram オプションの削除** データベース・サーバは、min_table_size_for_histogram オプションを使用しなくなりました。ソフトウェアの以前のバージョンでは、このオプションを使用して、ヒストグラムの作成対象となる最小テーブル・サイズを指定できました。現在のバージョンでは、Adaptive Server Anywhere は、5 つ以上のローから成るすべてのテーブルに対して、自動的にヒストグラムを作成します。CREATE STATISTICS 文を使用すると、サイズに関係なく、すべてのテーブルに対してヒストグラムを作成できます。

  「カラム統計の更新」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **廃止されたデータベース・オプション** 次のデータベース・オプションは、サポートされなくなりました。
  - truncate_date_values
  - assume_distinct_servers
- **古いデータベース・フォーマットの廃止** SQL Anywhere Studio の今後のメジャー・リリースでは、ソフトウェアの旧バージョンで作成されたデータベースはサポートされません。移行ツールは提供されます。
- **UNIX 用の非スレッド型 DBTools ライブラリの廃止** UNIX 用の非スレッド型 DBTools ライブラリ (*libdbtool9.so*) は廃止されます。現在のソフトウェアでは完全にサポートされていますが、SQL Anywhere Studio の今後のメジャー・リリースではサポートされません。
- **950TWN 照合の非サポート** 950ZHO_HK 照合と 950ZHO_TW 照合が 950TWN 照合より優先されます。

  「サポートされている照合と代替照合」『SQL Anywhere サーバ - データベース管理』を参照してください。

その他の動作の変更

ここでは、これまでのバージョンとは異なる動作をリストにして説明します。

- **トランザクション・ログを削除する場合のトランザクション・ログ・ユーティリティ (dblog) の制限** -n オプションを使用してトランザクション・ログを削除する場合は、対応するトランザクション・ログ・オフセットを無視するオプション (Log Transfer Manager の場合は -il、SQL Remote の場合は -ir、dbmlsync の場合は -is) も指定する必要があります。

  詳細については、「トランザクション・ログ・ユーティリティ (dblog)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **クワイエット・モードで動作するデータベース・ユーティリティ** -q オプション (クワイエット・モード) が指定された状態で次のいずれかの操作を実行する場合は、-y オプションも指定する必要があります。

  - サービス作成 (dbsvc) ユーティリティでサービスの変更または削除を行う
  - データ・ソース (dbdsn) ユーティリティでデータ・ソースの変更または削除を行う
  - 消去 (dberase) ユーティリティでファイルを消去する
  - ログ変換 (dbtran) ユーティリティでトランザクション・ログを変換する

- **ECC_TLS 暗号化または RSA 暗号化を使用する場合は証明書名とパスワードの指定が必要** 証明書のデフォルト値である certificate_password パラメータと trusted_certificates パラメータが削除されました。これらのデフォルト値は、SQL Anywhere Studio インストール環境の *win32* ディレクトリに提供されているサンプル証明書を利用していました。サンプル証明書は、テストと開発の目的でのみ効果的ですが、セキュリティは備えていません。

  また、-ec **all** サーバ・オプションもサポートされなくなりました。

  「-ec サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **-xs サーバ・オプションの変更** -xs **all** サーバ・オプションは、HTTP ポートと HTTPS ポートの両方で、Web 要求の受信がサポートされなくなりました。

  「-xs サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **デフォルト・ポートを使用していない場合、Mac OS X、HP-UX、Tru64 上のネットワーク・データベース・サーバに対して TCP/IP ポート番号を指定する必要がある** Mac OS X、HP-UX、または Tru64 でデータベース・サーバを起動する場合、デフォルト・ポート (2638) が使用中またはデフォルト・ポートを使用しないのであれば、ServerPort [PORT] プロトコル・オプションを使用してポート番号を指定する必要があります。

  「ServerPort プロトコル・オプション [PORT]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **アンロード・ユーティリティ (dbunload) でアンロードおよび再ロードされるデータベースの dbspace ファイル名が変更される** アンロード・ユーティリティ (dbunload) の -an オプションでデータベースをアンロードおよび再ロードした場合、新しいデータベースの dbspace ファ

イル名は、末尾に **R** が追加されます。これは、新しい dbspace ファイルが、元の dbspace ファイルと同じディレクトリに配置された場合の、名前の競合を回避するために行われます。Sybase Central の [データベース・アンロード] ウィザードを使用してデータをアンロードおよび再ロードした場合も、dbspace ファイル名には R が追加されます。

「アンロード・ユーティリティ (dbunload)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **LONG VARCHAR 値を返すプロパティ関数**　次の関数はこれまで VARCHAR(254) 値を返していました。現在は、VARCHAR(*maxpropsize*) 値を返します。ここで、*maxpropsize* は、サーバに対して指定されている最大ページ・サイズに基づきます。
  - CONNECTION_PROPERTY
  - DB_EXTENDED_PROPERTY
  - DB_PROPERTY
  - EVENT_PARAMETER
  - PROPERTY
- **STRTOUUID 関数の変更**　以前のリリースでは、STRTOUUID は、無効な UUID 値が渡されると NULL を返していました。現在は、conversion_error オプションが OFF に設定されていないかぎり、変換エラーを返します。conversion_error オプションが OFF の場合は、NULL を返します。

Mobile Link の動作の変更

ここでは、これまでのバージョンとは異なる動作をリストにして説明します。

セキュリティ動作の変更

- **HTTPS を使用するために廃止された HTTP+TLS セキュリティ**　HTTP で接続しているクライアントを対象とするトランスポート・レイヤ・セキュリティは廃止されました。HTTP でトランスポート・レイヤ・セキュリティを使用するには、HTTPS を使用する必要があります。

  サーバ側のセキュリティの詳細については、「トランスポート・レイヤ・セキュリティを使用する Mobile Link サーバの起動」『SQL Anywhere サーバ - データベース管理』を参照してください。

  クライアント側のセキュリティの詳細については、「トランスポート・レイヤ・セキュリティを使用する Mobile Link クライアントの設定」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Mobile Link で ECC_TLS 暗号化または RSA 暗号化を使用する場合は証明書名とパスワードの指定が必要**　証明書のデフォルト値である certificate_password 同期パラメータと trusted_certificates 同期パラメータが削除されました。これらのデフォルト値は、SQL Anywhere Studio インストール環境の *win32* ディレクトリに提供されているサンプル証明書を利用していました。サンプル証明書は、テストと開発の目的でのみ効果的ですが、セキュリティは備えていません。

「-x オプション」 『Mobile Link - サーバ管理』を参照してください。

Mobile Link のその他の動作の変更

- **UDP 受信のポーリング間隔がない** Listener の UDP 接続のポーリング間隔がなくなりました。Listener は、メッセージを即時に処理します。

  「Listener ユーティリティ」 『Mobile Link - サーバ起動同期』の **-l オプション**を参照してください。

- **Treo 180 Smartphone と Kyocera 6035 Smartphone における Mobile Link Palm Listener サポートの廃止** Palm Listener のサポート・デバイスの詳細については、「Palm デバイス用 Listener」 『Mobile Link - サーバ起動同期』を参照してください。

SQL Remote の動作の変更

非推奨機能とサポートされなくなった機能

ここでは、サポートを終了した機能、今後使用することを推奨しない機能の中で、既存のアプリケーションに影響する可能性があるものがリストされています。

- **SQL Remote for Adaptive Server Enterprise の廃止** SQL Anywhere Studio の今後のメジャー・リリースでは、SQL Remote for Adaptive Server Enterprise がなくなります。Mobile Link は、Adaptive Server Enterprise データベースと Adaptive Server Anywhere データベース間のデータ同期に対して、より柔軟でスケーラブルなソリューションを提供します。

その他の動作の変更

ここでは、これまでのバージョンとは異なる動作をリストにして説明します。

- **抽出 (dbxtract) ユーティリティ** dbxtract でリモート・データベースを抽出するときに -q オプション (クワイエット・モード) を指定する場合は、既存のコマンド・ファイルが確認されずに自動的に置換されるように、-y オプションも指定する必要があります。

  「抽出ユーティリティ」 『SQL Remote』を参照してください。

- **IPM_Receive メッセージ制御パラメータ** MAPI IPM_Receive メッセージ制御パラメータのデフォルト値が、YES に変更されました。この値を YES に設定すると、IPC メッセージと IPM メッセージの両方が SQL Remote に選択されます。

  「MAPI メッセージ・システム」 『SQL Remote』を参照してください。

Ultra Light の動作の変更

Ultra Light の今後のメジャー・リリースでは、業界標準 API を使用した開発が強化され、本来の静的型インタフェースに代わってコンポーネント・モデルを使用した開発が強化されます。これらの変更は、Ultra Light を使用したアプリケーション開発が簡単になるなど、ユーザにとってさまざまな利点をもたらします。

この計画の結果、このリリースでは各種の Ultra Light API が使用されなくなりました。つまり、現在のソフトウェアでは引き続き完全にサポートされていますが、今後のメジャー・リリースではサポートされません。今後のメジャー・リリースでは、旧式のインタフェースを使用するアプリケーションの移行を支援するツールが提供されます。

ここにリストされている非推奨機能とサポートされなくなった機能は、あくまでも予定であり、変更の可能性があります。

非推奨機能とサポートされなくなった機能

次の機能は、使用が推奨されないか、サポートが終了しました。

- **静的インタフェースの廃止** SQL Anywhere Studio の今後のメジャー・リリースでは、静的型 C++ API または静的型 Java API をサポートしません。Embedded SQL インタフェースは使用できますが、現在生成されているコード・メカニズムを通じては使用できません。
- **ADO.NET に置き換えられる Ultra Light.NET コンポーネント・インタフェース** このリリースでは、Ultra Light.NET は、新しい iAnywhere.Data.UltraLite ネームスペースでの ADO.NET 開発をサポートします。ADO.NET は、業界標準インタフェースの利点を備えており、大規模なアプリケーションを Adaptive Server Anywhere に簡単に移行することができます。Ultra Light.NET コンポーネント API (iAnywhere.UltraLite ネームスペース) は、このリリースでは使用が推奨されず、今後のメジャー・リリースでは提供されません。
- **JDBC に置き換えられる Native Ultra Light for Java コンポーネント・インタフェース** 現在の Native Ultra Light for Java インタフェースは、JDBC インタフェースに置き換えられることが予定されています。

その他の動作の変更

ここでは、これまでのバージョンとは異なる動作をリストにして説明します。

- **ダウンロード時の参照整合性に基づく削除に対して警告が出力される** Ultra Light は、ダウンロード時に参照整合性を維持するために、必要に応じてローを自動的に削除します。このように削除されたローごとに、警告が出力されるようになりました。

  「参照整合性と同期」『Mobile Link - クイック・スタート』を参照してください。
- **Native Ultra Light for Java の動作の変更** Cursor.getRowCount() メソッドが int を返すようになりました。ただし、アプリケーションの変更は必要ありません。
- **Ultra Light.NET コンポーネントの動作の変更** Cursor.getRowCount() メソッドが int を返すようになりました。ただし、アプリケーションの変更は必要ありません。
- **無効な同期パラメータの処理** 以前のリリースでは、Ultra Light ランタイムは、無効な同期パラメータはすべて無視していました。このため、スペルミスのパラメータは無視され、デフォルト値が代わりに使用されていました。

  このリリースでは、無効なパラメータが検出されると同期が失敗し、SQL コードの SQLE_UNRECOGNIZED_OPTION が設定されます。エラー・コールバックが設定されている場合は、無効なパラメータごとにそのエラー・コールバックが呼び出されます。重複は、引き続き無視されます。

- **セキュリティ機能を備えた同期のための新しいライブラリ**　同期用のセキュリティ・オプションが、別個のライブラリに移動されました。暗号化された同期で ULSecureCerticomTLSStream または ULSecureRSATLSStream セキュリティ・オプションを使用する場合は、対応する静的ライブラリに対して個別にリンクするか、個々の DLL を提供する必要があります。
- **Ultra Light for MobileVB と Crossfire の統合**　既存のプロジェクトで、ソフトウェアの以前のバージョンの Crossfire と統合されている Ultra Light for MobileVB を使用している場合は、Interop.UltraLiteAFLib.dll への参照を、iAnywhere.UltraLiteForAppForge.dll に変更する必要があります。

  「チュートリアル：AppForge Crossfire のサンプル・アプリケーション」『Ultra Light - AppForge プログラミング』を参照してください。

QAnywhere の動作の変更

非推奨機能とサポートされなくなった機能

- **QAnywhere Agent オプション**　次の QAnywhere Agent (qaagent) オプションが廃止され、置換されました。

廃止された qaagent オプション...	置換後の qaagent オプション...
-agent_id *id*	**-id** *id*
-dbauser *user*	**-c "UID=***user***"**
-dbeng *name*	**-c "ENG=***name***"**
-dbfile *filename*	**-c "DBF=***filename***"**
-dbname *name*	**-c "DBN=***name***"**
-ek *key*	**-c "DBKEY=***key***"**
-password *password*	**-c "PWD=***password***"**
-sv	**-c "Start={ dbeng9 | dbsrv9 }"**
-verbose	**-v**[*levels*]

また、次の qaagent オプションは不要となり、廃止されました。

- **-e**
- **-rb**

「qaagent 構文」『QAnywhere』を参照してください。

- **QAnywhere Agent はクライアント・メッセージ・ストアを作成しない**　メッセージ・ストア・データベースを各自で作成してから、qaagent を実行する必要があります。QAnywhere が必要とするシステム・オブジェクトでデータベースを初期化する、-si という新しいオプションが用意されています。

詳細については、「クライアント・メッセージ・ストアの設定」『QAnywhere』を参照してください。

- **InReplyToID が付いた QAnywhere メッセージ**　QAnywhere の組み込みヘッダ InReplyToID は、JMS プロパティ ias_ReplyToAddress にマッピングされなくなりました。つまり、JMSCorrelationID にコピーされなくなりました。

  「QAnywhere メッセージの JMS メッセージへのマッピング」『QAnywhere』を参照してください。

その他の動作の変更

- **QAnywhere クライアント・ライブラリ・メッセージのオーバヘッドの縮小**　通常のメッセージに対して、ias_MessageType プロパティは設定されなくなりました。ネットワーク・ステータスと、システム・キューに送信される他のシステム・メッセージに対しては引き続き設定されます。

第 4 章

バージョン 9.0.1 の新機能

目次

バージョン 9.0.1 の新機能

この項では、SQL Anywhere Studio バージョン 9.0.1 のコンポーネントに導入された新機能について説明します。

Adaptive Server Anywhere の新機能

この項では、Adaptive Server Anywhere バージョン 9.0.1 の新機能について説明します。新機能について主要なものからそうでないものまですべてを示し、各機能の詳細についての参照先も記述しています。

OLAP の強化

- **OLAP クエリの拡張**　一連のオンライン分析処理 (OLAP) 機能を使用して、データベース内のデータをより詳細に分析できます。強化機能には、CUBE と GROUPING SETS を使用して小計ローを結果セットに柔軟に追加する機能や、移動平均および他の高度な機能を提供する Window 関数などが含まれます。

  これらの OLAP 機能のサポートは、クエリ・エディタに組み込まれています。これらを使用して、ROLLUP、CUBE、GROUPING SETS 演算を使用するクエリを構築できます。

  次の項を参照してください。

  - 「OLAP のサポート」『SQL Anywhere サーバ - SQL の使用法』
  - 「ROLLUP の使用」『SQL Anywhere サーバ - SQL の使用法』
  - 「CUBE の使用」『SQL Anywhere サーバ - SQL の使用法』
  - 「GROUP BY GROUPING SETS」『SQL Anywhere サーバ - SQL の使用法』
  - 「Window 集合関数」『SQL Anywhere サーバ - SQL の使用法』

- **新しい統計関数**　複数の統計関数が追加されました。

  次の項を参照してください。

  - 「COS 関数 [数値]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「COVAR_POP 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「COVAR_SAMP 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「CUME_DIST 関数 [ランキング]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「DENSE_RANK 関数 [ランキング]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「PERCENT_RANK 関数 [ランキング]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「RANK 関数 [ランキング]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「REGR_AVGX 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「REGR_AVGY 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「REGR_COUNT 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「REGR_INTERCEPT 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「REGR_R2 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「REGR_SLOPE 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「REGR_SXX 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「REGR_SXY 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』

- 「REGR_SYY 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』
- 「ROW_NUMBER 関数 [その他]」『SQL Anywhere サーバ - SQL リファレンス』

- **新しい文字列関数** 次の文字列関数が追加されました。

  - 「BASE64_DECODE 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「BASE64_ENCODE 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「COMPRESS 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「DECOMPRESS 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「DECRYPT 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「ENCRYPT 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』
  - 「HASH 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』

文の強化

- **LOAD TABLE の強化** LOAD TABLE 文には、作成される統計を制限するための句が用意され、テーブルをより速くロードできるようになりました。また、この文には、ファイルの先頭の数行を無視するための SKIP オプションもあります。

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

- **SELECT ...INTO *base-table*** この新しい SELECT 構文は、ベース・テーブルを作成し、このテーブルにクエリのデータを入力します。

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

- **SQL 文の変数の拡張サポート** 一部の文は、複数のロケーションで定数と同じように変数を使用可能にすることによって、より柔軟に作成されています。これは、変数を宣言して使用できるストアド・プロシージャやバッチで特に役に立ちます。その結果、以前には、より煩雑な方法でのみ使用できた機能を EXECUTE IMMEDIATE で使用できます。

  次の文には、このような変数の拡張サポートがあります。

  - SELECT 文の TOP 句では、定数や整数変数が参照できます。「SELECT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - BACKUP 文 *backup-directory* と *archive-root*。「BACKUP 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - RESTORE 文 *filename*、*archive-root*、新しい *dbspace-name*。「RESTORE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - LOAD TABLE 文 *filename*。「LOAD TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - UNLOAD 文と UNLOAD TABLE 文 *filename*。「UNLOAD 文」『SQL Anywhere サーバ - SQL リファレンス』と「UNLOAD TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **SET 文の強化** SET 文は、Microsoft SQL Server との互換性のためにオプション ansi_nulls (ansinulls オプションと同じ) に対応できるようになりました。

詳細については、「SET 文 [T-SQL]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **ALTER TABLE 文の拡張** ALTER TABLE は、デフォルト値をもつ NOT NULL カラムを空でないテーブルに追加できるようになりました。この機能によって、既存のテーブルを変更するときの柔軟性が増しました。

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

- **ALTER VIEW 文の強化** ALTER VIEW 文は、基本となるテーブルのカラムを変更するときにビュー定義を再作成するための RECOMPILE 句をサポートできるようになりました。

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

- **MESSAGE 文の強化** MESSAGE 文には、FOR CONNECTION 句が追加されました。

  また、MESSAGE 文には、DEBUG ONLY 句も追加されました。debug_messages オプションを ON に設定すると、DEBUG ONLY 句を含む MESSAGE 文のあるすべてのストアド・プロシージャとトリガに対してデバッグ・メッセージが表示されます。

  「MESSAGE 文」『SQL Anywhere サーバ - SQL リファレンス』と「debug_messages オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

セキュリティの強化

- **データベース・ページ・チェックサム** データベース・ページ・チェックサムは、データベース・ページがディスクで修正されたかどうかを判断するのに使用します。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前に計算されます。ページがディスクから読み出されるときに、チェックサムは再計算されて、保存されているチェックサムと比較されます。値が異なる場合は、ページがディスクで変更されたか、破損しています。この機能を使用するには、チェックサムを有効にして既存のデータベースをアンロードしてからデータベースに再ロードします。データベースのチェックサムが有効かどうかを確認するには、Checksum プロパティを使用します。

  チェックサムを有効にしてデータベースを作成する方法については、「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』、「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバ - データベース管理』、「データベース・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

  また、チェックサムを使用してデータベースを検証することもできます。「VALIDATE 文」『SQL Anywhere サーバ - SQL リファレンス』、「検証ユーティリティ (dbvalid)」『SQL Anywhere サーバ - データベース管理』、「sa_validate システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

パフォーマンスの強化

複雑なクエリを含む広範なタスクのパフォーマンスを向上させるために多くの強化が行われています。これらの強化の一部は、単に内部的な強化です。その他の強化は、次のとおりです。

- **並列インデックス・スキャン** ハードウェアまたはソフトウェア RAID アレイなど、複数のディスクのスピンドルがあるボリュームでは、クエリ・オプティマイザは、インデックスを並列的に使用してテーブルをスキャンできるようになりました。

「並列インデックス・スキャン」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **クラスタード・ハッシュ GROUP BY アルゴリズム** Adaptive Server Anywhere クエリ・オプティマイザは、パフォーマンスを向上させるために、HAVING 句がローのごく一部を返す GROUP BY クエリの特定のクラスにとって特に役に立つ新しいアルゴリズムを使用できます。

  「クラスタード・ハッシュ GROUP BY アルゴリズム」『SQL Anywhere サーバ - SQL の使用法』と「optimization_workload オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **データベース・サーバのキャッシュ・ウォーミング** キャッシュ・ウォーミングをサポートするために、3 つの新しいデータベース・サーバのコマンド・ライン・オプションが追加されました。データベースが最後に起動したときに参照したデータベース・ページとともにデータベース・サーバのキャッシュを事前にロードすることによって、データベースに対して実行される初期クエリの実行時間を短縮できるよう、キャッシュ・ウォーミングが設計されています。キャッシュ・ウォーミングを使用することによって、データベースが起動するたびにデータベースに対して同じクエリが実行されるときのパフォーマンスを向上させることができます。

  「-cc サーバ・オプション」『SQL Anywhere サーバ - データベース管理』、「-cr サーバ・オプション」『SQL Anywhere サーバ - データベース管理』、「-cv サーバ・オプション」『SQL Anywhere サーバ - データベース管理』、「キャッシュ・ウォーミングの使用」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **オプティマイザに関するヒント** WITH (XLOCK) は、FROM 句の新しいテーブル・ヒント機能です。XLOCK は、ヒントが指定されたテーブルの文によって処理されるローを排他的にロックすることを指定します。影響を受けるローは、トランザクションの終わりまでロックされたままになります。この機能はすべての独立性レベルで動作します。

  WITH INDEX ヒントは、クエリの最適化中にオプティマイザに特定のインデックスを使用させます。これは高度な機能で、正しく使用しないとパフォーマンスが低下する可能性があります。このため、この機能は経験のあるユーザのみ使用してください。

  「FROM 句」『SQL Anywhere サーバ - SQL リファレンス』と「頻繁に検索するカラムにインデックスを使う」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **NetWare の内部実行スレッドのデフォルトのスタック・サイズの増加** NetWare の内部実行スレッドのデフォルトのスタック・サイズは、128 KB に増加されました。

  「-gss サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

プログラミング・インタフェースの強化

- **Perl インタフェース** Perl DBI モジュールの Perl の新しい DBD::ASAny ドライバを使用して、Perl スクリプトから Adaptive Server Anywhere データベースにアクセスして修正できます。

- **InstallShield プロジェクト** SQL Anywhere Studio には、InstallShield Merge Module Projects と Object Projects が組み込まれました。これらのプロジェクトを使用して Merge Modules と Objects を生成し、現在コンピュータにインストールされているソフトウェアを再配備できます。SQL Anywhere の前のバージョンには、Merge Modules と Objects が含まれています。これ

らを使用して元のソフトウェアを再配備することもできますが、EBF を適用した後の再配備としては便利な手段ではありません。

管理の強化

- **BACKUP の強化**　BACKUP 文には、イメージのバックアップ用として ON EXISTING ERROR 句が組み込まれました。この句を指定すると、バックアップがすでに存在するときにファイルを作成するとエラーが発生します。

  BACKUP 文のアーカイブのバックアップが、以前はイメージのバックアップでのみ使用可能だったオプションをサポートできるよう強化されています。

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

  バックアップ・ユーティリティは、サーバ・コンピュータ上でバックアップを作成できるようになりました。以前は、バックアップはクライアント・コンピュータ上でのみ作成できました。

  「バックアップ・ユーティリティ (dbbackup)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **アンロード・ユーティリティ (dbunload) の強化**　アンロード・ユーティリティは、データベースをアンロードするときにビューの依存性を自動的に処理できるようになりました。ソフトウェアの以前のバージョンで、ビュー定義を *reload.sql* ファイルに複数回出力するために使用されていた -j オプションは、使用されなくなりました。アンロード・ユーティリティは、他のビューに依存するビュー定義のアンロードを自動的に処理します。

  また、アンロード・ユーティリティを使用して、新しいデータベースにアンロードするときのデータベース・ページ・サイズを変更できます。

  「アンロード・ユーティリティ (dbunload)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **サーバ列挙ユーティリティ (dblocate) の強化**　サーバ列挙 (dblocate) ユーティリティでは、ホスト名または IP アドレスを指定し、データベース・サーバの検索対象を特定のコンピュータに制限できるようになりました。また、IP アドレスがコンピュータ名に解析されないよう指定する -n オプションをサポートできるようになったため、パフォーマンスが向上します。

  「サーバ列挙ユーティリティ (dblocate)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Adaptive Server Anywhere コンソール・ユーティリティによる統合化ログインのサポート**　Windows NT/2000/XP で Adaptive Server Anywhere コンソール (dbconsole) ユーティリティに接続する場合、[接続] ダイアログでは、統合化ログインを使用してデータベースに接続できます。

  「[接続] ダイアログ：[ID] タブ (SQL Anywhere の場合)」『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

- **データベース・サーバを再起動せずに要求ログ・ファイルを変更可能**　データベース・サーバの起動時には、-zs サーバ・オプションを使用して、要求ログ・ファイルのサイズを指定できます。sa_server_option システム・プロシージャを使用すると、データベース・サーバを再起動せずに、要求ログ・ファイルのサイズを変更できます。

「sa_server_option システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **プロファイリング・システム・トリガに追加された追加情報** sa_procedure_profile システム・プロシージャと sa_procedure_profile_summary システム・プロシージャは、データベースでプロシージャ・プロファイリングがオンになっている場合、システム・トリガに関する追加情報を返すようになりました。

  「sa_procedure_profile システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』と「sa_procedure_profile_summary システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **新しいシステム・テーブル** データベースが起動されたプラットフォームとソフトウェアの異なるバージョンに関する情報を管理する新しいシステム・テーブルが追加されました。

  「ISYSHISTORY システム・テーブル」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **新しい照合** リトアニア語 (1257LIT、ANSI コード・ページ 1257) をサポートする照合、トルコ語 (1254TRKALT) をサポートする照合の 2 つの照合が使用可能になりました。トルコ語の照合では、I-dot と I-no-dot は区別されません。

  「サポートされている照合と代替照合」『SQL Anywhere サーバ - データベース管理』と「代替トルコ語照合 1254TRKALT」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **dedicated_task オプション** このオプションを指定すると、要求処理タスクは 1 つの接続からの要求の処理専用になります。この事前に確立された接続を使用すると、データベース・サーバが応答しなくなったときにこのサーバのステータスに関する情報を収集できます。

  「dedicated_task オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

Interactive SQL の強化

- **Interactive SQL を使用すると、ファイルの読み込みと書き込みに使用されるコードを指定できます。** Interactive SQL の READ、INPUT、OUTPUT 文は、ファイルの読み込みと書き込みに使用される文字コードを指定できるオプションのコード句をサポートできるようになりました。default_isql_encoding オプションが追加されることで、後続の READ、INPUT、OUTPUT 文に使用される文字コードを指定できます。

  「default_isql_encoding オプション [Interactive SQL]」『SQL Anywhere サーバ - データベース管理』、「READ 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』、「INPUT 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』、「OUTPUT 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  また、Interactive SQL の [インポート] ウィザードと [エクスポート] ウィザードを使用するときにファイルの読み込みと書き込みに使用される文字コードも指定できます。

  「データのインポートとエクスポート」『SQL Anywhere サーバ - SQL の使用法』と「[エクスポート] ダイアログ」『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

- **Interactive SQL による統合化ログインのサポート** Windows NT/2000/XP で Interactive SQL に接続する場合、[接続] ダイアログでは、統合化ログインを使用してデータベースに接続できます。

  「[接続] ダイアログ：[ID] タブ (SQL Anywhere の場合)」『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

- **InteractiveSQL では、結果セットの表示に使用されるフォントを設定できます。** Interactive SQL の [結果] ウィンドウ枠内に表示されるデータのフォント、フォント・スタイル、ポイント・サイズを選択できます。

  Interactive SQL の [結果] ウィンドウ枠の設定については、「[オプション] ダイアログ：[結果] タブ」『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

- **Interactive SQL を使用すると、ファイルのブラウズに使用される初期フォルダを指定できます。** Interactive SQL でファイルをブラウズする場合、初期ディレクトリとして現在のディレクトリ (オペレーティング・システムによって定義されているディレクトリ) と、ファイルが最後に開かれたフォルダのどちらを使用するかを指定できます。

  「[オプション] ダイアログ：[一般] タブ」『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

Sybase Central の強化

- **Sybase Central では、結果セットの表示に使用されるフォントを設定できます。** テーブルを選択するときに、Sybase Central の [データ] タブに表示されるデータのフォント、フォント・スタイル、ポイント・サイズを選択できます。

  Sybase Central の [データ] タブの設定については、「[プラグインの環境設定] ダイアログ：[テーブル・データ] タブ」『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

- **[リモート・サーバ作成] ウィザードが現在のユーザの外部ログインの作成をサポート** [リモート・サーバ作成] ウィザードでは、現在のユーザの外部ログインを作成することによって、リモート・サーバを作成する前に外部ログインを作成する必要がなくなりました。

  「Sybase Central を使用したリモート・サーバの作成」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **Sybase Central による統合化ログインのサポート** Windows NT/2000/XP で Sybase Central に接続する場合、[接続] ダイアログでは、統合化ログインを使用してデータベースに接続できます。

  「[接続] ダイアログ：[ID] タブ (SQL Anywhere の場合)」『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

- **Sybase Central の [ビュー] メニューを使用してカラムをソート可能** Sybase Central の [ビュー] メニューの [ソート] 項目を使用すると、右ウィンドウ枠にあるカラム見出しをクリックする代わりに、右ウィンドウ枠でカラムをソートできます。

- **[外部キー] プロパティ・シートから外部キー設定を変更可能** Sybase Central の外部キー設定は、[外部キー] プロパティ・シートから変更できます。

「[設定の変更] ダイアログ」『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

- **[プロキシ・テーブル] ウィザードでプライマリ・キー・カラム情報を表示可能** 以前は、[プロキシ・テーブル] ウィザードを使用してプロキシ・テーブルを作成する場合、リモート・テーブルのプライマリ・キーに属しているカラムを確認できませんでした。しかし、プライマリ・キーに属しているカラムをウィザードで確認できるようになりました。
- **ユーティリティ・ウィザードをキャンセル可能** [データベース・アップグレード] ウィザード、[データベース・バックアップ] ウィザード、[データベース・リストア] ウィザード、[データベース検証] ウィザード、[データベース圧縮] ウィザード、[データベース展開] ウィザード、および [バックアップ・イメージ作成] ウィザードは、キャンセルできます。これらのウィザードには、操作が成功したか失敗したかについてのステータス情報が表示されるメッセージ・ダイアログがあります。
- **Sybase Central はサービスの作成と編集時に domain\user の形式のアカウント名をサポート** [サービス作成] ウィザードと [サービス] プロパティ・シートでは、サービスの作成と編集時に **domain¥user** の形式のアカウント名を入力できるようになりました。アカウント名は、[サービス] プロパティ・シートまたは [サービス作成] ウィザードの [アカウント] タブの [その他のアカウント] フィールドに入力できます。

その他の機能強化

- **データベース・サーバは Linux プラットフォーム上で非同期 I/O を使用** Linux 上でデータベース・サーバを実行するときに、データベース・サーバは可能であれば、デフォルトで非同期 I/O を使用します。-ua データベース・サーバ・オプションでは、非同期 I/O の使用をオフにできます。

  「-ua サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **openxml による等号述部のサポート** openxml 関数では、XPath 式で等号述部を使用できます。この機能によって、属性値を使用して XML ドキュメント内のノードを検索できます。

  「openxml システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **TransactionStartTime 接続プロパティ** このプロパティは、COMMIT または ROLLBACK の後にデータベースが最初に修正された時間を返します。

  「接続レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **UserAppInfo プロパティ** このプロパティは、AppInfo 接続パラメータで指定された接続文字列の一部を返します。

  「接続レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **ConsoleLogFile サーバ・プロパティ** -o オプションを指定すると、このプロパティは、サーバ・メッセージ・ウィンドウからのメッセージがログに記録されるファイル名を返します。

「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **UNIX プラットフォームの DriveType データベース・プロパティ** DriveType データベース・プロパティは、UNIX プラットフォームに対応できるよう拡張されています。

  「データベース・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **接続 ID は 1 から始まり、データベース・サーバへの新しい接続ごとに 1 ずつ増加** データベース・サーバが起動すると、サーバへの接続ごとに 1 から始まる接続 ID が割り当てられます。この接続番号は、サーバへの新しい接続ごとに 1 ずつ増加します。接続 ID は、-z サーバ出力と LogFile 接続パラメータ出力のログに記録されます。また、接続 ID は、CONNECTION_PROPERTY、NEXT_CONNECTION、NEXT_DATABASE、および DROP CONNECTION 関数と要求のロギングによっても使用されます。

  「CONNECTION_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』、「NEXT_CONNECTION 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』、「NEXT_DATABASE 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』、「要求ロギング」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **NetWare と UNIX 上でのキャッシュ管理の向上** -c を使用して指定されたキャッシュ・サイズが UNIX または NetWare 上で使用可能なメモリ量よりも大きい場合、データベース・サーバは、使用可能なメモリ量に基づいて最大キャッシュ・サイズを計算できるようになりました。

  3 つの環境でのデータベース・サーバによる最大キャッシュ・サイズの計算方法については、「-c サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **odbc_distinguish_char_and_varchar オプション** odbc_distinguish_char_and_varchar オプションは、Adaptive Server Anywhere ODBC ドライバによる CHAR カラムの記述方法を制御します。

  「odbc_distinguish_char_and_varchar オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

Mobile Link の新機能

次に、バージョン 9.0.1 で導入したソフトウェアに加えられた変更と追加を示します。

- **QAnywhere メッセージ** Mobile Link QAnywhere には、アプリケーション対アプリケーションのメッセージ機能が用意されています。この機能によって、Windows または Windows CE オペレーティング・システムで動作しているさまざまなデバイス上のリモート・アプリケーションとメッセージを交換するアプリケーションを作成できます。

  QAnywhere『QAnywhere』を参照してください。

- **外部認証** Mobile Link のユーザ認証が強化され、LDAP サーバと POP3 電子メール・サービスなどの他の外部ソースを使用してユーザを簡単に認証できるようになりました。

  「外部サーバに対する認証」『Mobile Link - クライアント管理』を参照してください。

- **新しい Mobile Link システム・テーブル**　新しい Mobile Link システム・テーブルは複数あります。新しい Mobile Link システム・テーブルが存在するということは、Adaptive Server Anywhere データベースをアップグレードし、その他の統合データベースのアップグレード・スクリプトを実行する必要があります。

  「Mobile Link サーバ・システム・テーブル」『Mobile Link - サーバ管理』を参照してください。

- **設定可能なスクリプト・バージョン**　新しい ml_property Mobile Link システム・テーブルを使用すると、スクリプト・バージョンのプロパティを格納できます。

  「ml_property」『Mobile Link - サーバ管理』と「ml_add_property」『Mobile Link - サーバ管理』を参照してください。

- **iAnywhere ODBC ドライバ**　Windows で DB2 対応の iAnywhere ODBC ドライバを使用できるようになりました。これは Wire Protocol ドライバであるため、DB2 クライアント・ソフトウェアは必要ありません。

- **IBM DB2 設定スクリプトのバージョン番号**　Mobile Link サーバの複数のバージョンが同じ DB2 サーバ・インスタンスを使用できるようになりました。これは、Mobile Link が DB2 ストアド・プロシージャに対して使用する 2 つの Java クラスに SQL Anywhere Studio バージョン番号が使用されるようになったためです。9.0.1 リリースでは、これらは *SyncDB2_901.class* と *SyncDB2Long_901.class* と呼ばれます。

  「IBM DB2 UDB 統合データベース」『Mobile Link - サーバ管理』を参照してください。

- **新しい -us サーバ・オプション**　新しいサーバ・オプションを使用すると、Mobile Link が不要なテーブル・スクリプトを呼び出せなくなるため、パフォーマンスが向上します。

- **ActiveSync プロバイダによるアクティビティ・ログ・ファイルの生成**　ActiveSync プロバイダは、アクティビティのログを生成できるようになりました。

  「ActiveSync プロバイダ・インストール・ユーティリティ [mlasinst]」『Mobile Link - クライアント管理』を参照してください。

Adaptive Server Anywhere クライアントの強化

- **Adaptive Server Anywhere クライアントのアプリケーション統合の向上**　dbmlsync の新しい統合コンポーネントを使用すると、Windows プラットフォーム上で Adaptive Server Anywhere リモート・データベースと統合するアプリケーションをより簡単でカスタマイズしやすい方法で作成できます。

  「dbmlsync 統合コンポーネント」『Mobile Link - クライアント管理』を参照してください。

- **失敗したダウンロードの再開**　Adaptive Server Anywhere と Ultra Light の両方のリモート データベースでダウンロードが失敗したときに、長時間のデータ再送を防ぐことができるようになりました。ダウンロードの部分的再送の後、失敗したダウンロードを再開できる場合があります。

  「失敗したダウンロードの再開」『Mobile Link - サーバ管理』を参照してください。

- **Adaptive Server Anywhere クライアントのトランザクション・レベルのアップロード**　リモート・データベース上のトランザクションをアップロードに保持できるようになりました。また、この処理は同期単位で行うことができます。

  「-tu オプション」『Mobile Link - クライアント管理』を参照してください。

- **拡張オプションの新しい指定方法**　sp_hook_dbmlsync_set_extended_options という新しいフックを使用すると、次に行われる同期の動作をプログラムでカスタマイズできます。

  「sp_hook_dbmlsync_set_extended_options」『Mobile Link - クライアント管理』を参照してください。

サーバ起動同期

- **自動デバイス追跡**　自動デバイス追跡によって、通知プロセスが簡単になります。

  「デバイス・トラッキング」『Mobile Link - サーバ起動同期』を参照してください。

- **Sybase Central の設定**　Sybase Central Mobile Link プラグインを使用して、通知を設定できるようになりました。

  「プロパティの設定」『Mobile Link - サーバ起動同期』を参照してください。

- **オプションの配信確認**　メッセージを受信したときに確認が自動的に統合データベースに送信されるよう通知を設定できるようになりました。

  「Listener 構文」『Mobile Link - サーバ起動同期』を参照してください。

- **新しい Listener オプション**　Listener には、複数のチャネルを一度に受信する機能など、複数の新しいコマンド・ライン・オプションが用意されています。

  「Listener ユーティリティ」『Mobile Link - サーバ起動同期』を参照してください。

- **Palm の設定**　Palm デバイスを簡単に設定するための Palm Listener 設定ユーティリティ (dblsncfg) が用意されました。

  「Palm デバイス用 Listener」『Mobile Link - サーバ起動同期』を参照してください。

Mobile Link モニタ

- **モニタ・データのエクスポート**　モニタ・データをリレーショナル・データベースまたは Excel ファイルにエクスポートできるようになりました。

- **モニタの強化情報**　テーブルにどのカラムを表示するかをカスタマイズできるようになりました。また、以前は [同期] プロパティ・シートからのみ使用可能だった情報が含まれる新しいカラムや、モニタ・セッションの同期をユニークに識別する新しいカラムが用意されました。

- **ソート機能の向上**　モニタ・テーブルのソート機能が向上しました。データが追加または更新されるときのソート順を管理できるようになりました。

- **ユーザ・インタフェースの強化**　チャート内の選択項目をズーム・イン、ズーム・アウト、またはズームするための新しいメニューとツール・バー・ボタンが用意されました。また、一時停止によってチャートを自動的にスクロールするかどうかを制御できるようになりました。

「Mobile Link モニタ」『Mobile Link - サーバ管理』を参照してください。

新しい Web サーバのサポート

- **リダイレクタによる Apache Web サーバのサポート** Apache Web サーバ用の新しいネイティブなリダイレクタが用意されています。Apache Web サーバを使用する場合、Tomcat は必要ありません。

  「リダイレクタによる Web サーバを経由した同期」『Mobile Link - サーバ管理』を参照してください。

Ultra Light の新機能

Ultra Light 9.0.1 には、次の新機能が導入されています。

- **Ultra Light.NET コントロール** コントロールのセットが Visual Studio.NET 2003 ツールボックスに追加され、接続パラメータを指定しやすくなったり、Ultra Light.NET アプリケーションから同期をモニタしやすくなります。

  「チュートリアル：Ultra Light.NET アプリケーションの構築」『Ultra Light - .NET プログラミング』を参照してください。

- **Ultra Light for M-Business Anywhere** 以前は AvantGo M-Business Server と呼ばれていた新しいコンポーネントが iAnywhere M-Business Anywhere で使用できます。

  Ultra Light - M-Business Anywhere プログラミング『Ultra Light - M-Business Anywhere プログラミング』を参照してください。

- **動的 SQL の CREATE と DROP 文** CREATE/DROP TABLE と CREATE/DROP INDEX 文は、動的 SQL で使用できるようになりました。Ultra Light コンポーネントのユーザは、これらの文を使用して、Ultra Light データベースのスキーマを変更できます。

  「Ultra Light CREATE INDEX 文」『Ultra Light - データベース管理とリファレンス』と「Ultra Light CREATE TABLE 文」『Ultra Light - データベース管理とリファレンス』を参照してください。

- **動的 SQL からのトランザクションの制御** COMMIT と ROLLBACK 文は、動的 SQL で使用できるようになりました。Ultra Light コンポーネントのユーザは、これらの文によって、SQL 文を使用してトランザクションを制御できます。これらの文は、接続オブジェクトに対してコミットとロールバック・メソッドの代わりとして使用できます。

  「Ultra Light COMMIT 文」『Ultra Light - データベース管理とリファレンス』と「Ultra Light ROLLBACK 文」『Ultra Light - データベース管理とリファレンス』を参照してください。

- **動的 SQL SELECT の強化** WHERE 句または HAVING 句の探索条件でサブクエリを使用できます。これらは、FROM 句の抽出テーブルとしても使用できます。

  「Ultra Light SELECT 文」『Ultra Light - データベース管理とリファレンス』と「Ultra Light の探索条件」『Ultra Light - データベース管理とリファレンス』を参照してください。

  HAVING 句がサポートされるようになりました。「Ultra Light SELECT 文」『Ultra Light - データベース管理とリファレンス』を参照してください。

- **Ultra Light に対する ODBC インタフェース**　Ultra Light は、ODBC プログラミング・インタフェースのサブセットをサポートできるようになりました。
- **C++ インタフェースの混在**　Ultra Light C/C++ ベースのインタフェース (Embedded SQL、静的型 C++ API、C++ コンポーネント) は、同じアプリケーションで使用できます。

  中でも注目すべきは、C++ コンポーネントの動的 SQL が既存の Embedded SQL または静的型 C++ API アプリケーションに追加されたことや、Embedded SQL を使用して SQL を主に C++ コンポーネント・ベースのアプリケーションで実行できることです。

  新機能には、「db_start_database 関数」『Ultra Light - C/C++ プログラミング』(Embedded SQL) と StartDatabase メソッドが含まれています。
- **Ultra Light C++ コンポーネントの CodeWarrior ステーショナリ**　ステーショナリ **Palm OS Ultra Light C++ コンポーネント・アプリケーション**は、Ultra Light plug-in for CodeWarrior の一部として提供されています。これは、Palm OS 用の CodeWarrior を使用した C++ コンポーネント・アプリケーションの構築を支援します。

  Ultra Light plug-in for CodeWarrior のファイルは、Ultra Light のインストール中にディスクにコピーされますが、プラグインは、追加のインストール手順を実行しなければ使用できません。

  「Metrowerks CodeWarrior で Ultra Light アプリケーションを開発する」『Ultra Light - C/C++ プログラミング』を参照してください。
- **Ultra Light C/C++ のエラー処理の向上**　エラー・コールバックは、すべての Ultra Light C/C++ インタフェースでサポートされるようになりました。コールバックを使用すると、アプリケーションにすべてのエラーを通知できるため、開発者は開発時に貴重な情報を得ることができます。

  「ULRegisterErrorCallback 関数」『Ultra Light - C/C++ プログラミング』と「ULRegisterErrorCallback のコールバック関数」『Ultra Light - C/C++ プログラミング』を参照してください。
- **Ultra Light コンポーネントはエンジンを使用可能**　複数のアプリケーションからの接続を受け入れられる Ultra Light データベース・エンジンは、Ultra Light コンポーネントの代替導入オプションとして使用できるようになりました。

  このオプションは、Ultra Light for MobileVB または Ultra Light ActiveX には使用できません。
- **データベース変換ツール**　ulconv ユーティリティは、Ultra Light データベース上で多くの処理を行うためのコマンド・ライン・ツールです。これには、XML ファイルへのデータベースのアンロード、XML ファイルからの新しいデータベースのロード、データベース・フォーマットの変換などの処理があります。
- **同期プログレス・イベントの追加**　エラーが発生し、ダウンロードされた変更内容がロールバックされる場合、同期オブザーバは追加イベントを使用できます。

  使用している API の ULSyncState 構造体またはオブジェクトを参照してください。
- **スキーマのアップグレードの監視**　スキーマのアップグレードには時間がかかることがあります。新しいスキーマのアップグレード・イベントには、アプリケーションがスキーマのアップグレードの進行状況を監視するためのメカニズムが採用されています。

- **再起動可能なダウンロード**　Ultra Light は、同期 observer を介して、通信エラーやユーザによるキャンセルが原因で失敗したダウンロードを再起動できるようになりました。

  「失敗したダウンロードの再開」『Mobile Link - サーバ管理』を参照してください。

- **新しい Windows CE プラットフォームのサポート**　Ultra Light は、Smartphone 2002 プラットフォームをサポートできるようになりました。このプラットフォームでは、ActiveSync 同期はサポートされません。

  Ultra Light は、V4T ("thumb") モードの ARM チップ上の Windows CE 4.1 もサポートしています。

- **マルチデータベースのサポート**　Ultra Light コンポーネントは、異なるデータベース・ファイル名または作成者 ID を指定して複数の接続要求を発行することによって、1 つのアプリケーションから複数のデータベースに対応できます。

  この機能のため、接続パラメータにはいくつかの拡張機能があります。

  「Ultra Light DBN 接続パラメータ」『Ultra Light - データベース管理とリファレンス』を参照してください。

- **ULPalmLaunch と ULPalmExit は不要**　Ultra Light は、アプリケーションを閉じるときのステータス管理をサポートしやすくする追加の接続関連プリミティブをサポートできるようになりました。これらの新しい機能のため、Palm OS アプリケーションには、ULPalmLaunch や ULPalmExit などの Palm 固有の特別なプリミティブが必要なくなりました。

  「Ultra Light Palm アプリケーションのステータスの管理」『Ultra Light - C/C++ プログラミング』、「Ultra Light Palm アプリケーションのステータスの管理」『Ultra Light - AppForge プログラミング』、「データベースへの接続」『Ultra Light - C/C++ プログラミング』を参照してください。

- **Ultra Light データベース・プロパティ**　Ultra Light データベースのプロパティは、Ultra Light コンポーネント・アプリケーションで使用できるようになりました。大文字と小文字の区別、照合、およびグローバル・オートインクリメントの値に使用されるデータベース ID はすべて、API に応じて Connection オブジェクトのプロパティまたはメソッドとして使用できます。

マニュアルの強化

この項では、バージョン 9.0.1 の Adaptive Server Anywhere マニュアルの体裁、編成、またはナビゲーションの強化について説明し、主な変更についてはすべて説明します。

新しいマニュアル

- **Mobile Link サーバ起動同期**　Mobile Link サーバ起動同期機能は、適切なマニュアルに移動されました。

  Mobile Link - サーバ起動同期『Mobile Link - サーバ起動同期』を参照してください。

- **QAnywhere メッセージ**　Mobile Link の新しいメッセージ・アプリケーションである QAnywhere について説明する新しいマニュアルが作成されました。

  QAnywhere『QAnywhere』を参照してください。

- **新しい Mobile Link チュートリアル**　Java および .NET スクリプト論理を Adaptive Server Anywhere リモート データベースで使用する方法について説明する新しいチュートリアルが作成されました。

  「チュートリアル：Java 同期論理の使用」『Mobile Link - クイック・スタート』と「チュートリアル：.NET 同期論理の使用」『Mobile Link - クイック・スタート』を参照してください。

- **dbmlsync 統合コンポーネントについて説明する新しい章**　「dbmlsync 統合コンポーネント」『Mobile Link - クライアント管理』を参照してください。

- **Ultra Light C/C++ マニュアルの統合**　Ultra Light C/C++ インタフェース (Embedded SQL、静的型 C++ API、および C++ コンポーネント) のマニュアルは、ODBC インタフェースとともに 1 冊のマニュアルに統合されました。

  Ultra Light - C/C++ プログラミング『Ultra Light - C/C++ プログラミング』を参照してください。

- **Ultra Light for M-Business Anywhere**　新しい Ultra Light for M-Business Anywhere コンポーネントには独自のマニュアルが用意されました。

  Ultra Light - M-Business Anywhere プログラミング『Ultra Light - M-Business Anywhere プログラミング』を参照してください。

マニュアルの強化

- **『Adaptive Server Anywhere 入門』と『SQL Anywhere Studio の紹介』の統合**　『Adaptive Server Anywhere 入門』と『SQL Anywhere Studio の紹介』は 1 冊のマニュアルに統合されました。新しいマニュアルは、『SQL Anywhere Studio の紹介』と呼ばれます。

- **PDF マニュアルの強化**　一部のユーザにとって、PDF は、特に概念的なマニュアルの場合、HTML ベースのオンライン・マニュアルの代わりとして役に立ちます。オンライン・マニュアルの PDF バージョンは、現在、デフォルトでインストールされており、Windows の [スタート] メニューまたはオンライン・マニュアルから各トピックの上部の PDF 項目をクリックすることによってアクセスできます。

  PDF ファイルのクリッカブル機能は、マニュアル内だけでなく、他のマニュアルや Web サイト上の情報にもリンクしています。これらの機能の動作は、ファイルをブラウザのプラグインから読み込むか、または Acrobat Reader から直接読み込むかによって異なります。最善の方法は、Acrobat Reader を直接使用する方法です。

- **ヘルプに関するヘルプ**　新しい項では、Windows の HTML ヘルプ・フォーマットのオンライン・マニュアルと PDF フォーマットのオンライン・マニュアルの違いについて説明します。また、この項では、さまざまなヘルプ機能を使用してマニュアル内をナビゲーションし、目的の情報にアクセスする方法についても説明します。

バージョン 9.0.1 での動作の変更

この項では、SQL Anywhere Studio バージョン 9.0.1 のコンポーネントに導入された動作の変更について説明します。

Adaptive Server Anywhere の動作の変更

非推奨機能とサポートされなくなった機能

ここでは、サポートを終了した機能、今後使用することを推奨しない機能の中で、既存のアプリケーションに影響する可能性があるものがリストされています。

- **MDSR 暗号化の廃止**　以前は、Adaptive Server Anywhere では、強力な暗号化として MDSR と AES の両方がサポートされていました。現在、強力な暗号化としては唯一 AES 暗号化がサポートされています。この変更は、初期化ユーティリティ、抽出ユーティリティ、またはアンロード・ユーティリティで暗号化を使用する場合、**-ea** オプションが必要ないことを意味します。また、Create Database 文と Create Encrypted File 文の両方から *algorithm* パラメータが削除されます。
- **ライト・ファイルの廃止**　このリリースでは、ライト・ファイルの使用は推奨しません。
- **圧縮データベース・ファイルの廃止**　このリリースでは、圧縮データベース・ファイルの使用は推奨しません。
- **アンロード・ユーティリティ (dbunload) の -j オプションの廃止**　アンロード・ユーティリティが強化された結果、アンロード・ユーティリティの -j オプションはサポートされなくなりました。
- **言語選択ユーティリティ (dblang) の -d オプションの廃止**　Adaptive Server Anywhere のレジストリ設定を変更するために使用されていた言語選択ユーティリティの -d オプションは、サポートされなくなりました。

その他の動作の変更

ここでは、これまでのバージョンとは異なる動作をリストにして説明します。

- **CURRENT_TIMESTAMP と CURRENT_USER 特別値の追加**　Microsoft SQL Server との互換性を保つために、CURRENT_TIMESTAMP (CURRENT TIMESTAMP と同じ) と CURRENT_USER (CURRENT USER と同じ) 特別値が追加されました。

  「CURRENT TIMESTAMP 特別値」『SQL Anywhere サーバ - SQL リファレンス』と「CURRENT USER 特別値」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **データベース・サーバがすでに実行している場合、db_start_engine は SQLCODE 0 を返す**　db_start_engine は、データベース・サーバがすでに実行している場合、0 以外の値を返し、SQLCODE を 0 に設定するようになりました。以前は、db_start_engine は 0 以外の値を返しましたが、SQLCODE は SQLE_ENGINE_ALREADY_RUNNING に設定されていました。

  「db_start_engine 関数」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **データベースがすでに実行している場合、db_start_database は 0 以外の値と SQLCODE 0 を返す**　db_start_database は、データベースがすでに実行している場合、0 以外の値を返し、SQLCODE を 0 に設定するようになりました。以前は、db_start_database は 0 (失敗を示す) を返し、SQLCODE を SQLE_ALIAS_CLASH に設定していました。

  「db_start_database 関数」『SQL Anywhere サーバ - プログラミング』と「db_start_database 関数」『Ultra Light - C/C++ プログラミング』を参照してください。

- **検証ユーティリティのデフォルト・アルゴリズムの変更**　検証ユーティリティ (dbvalid) は、エクスプレス・チェック (-fx オプション) アルゴリズムをデフォルトで使用できるようになりました。このエクスプレス・チェック・アルゴリズムによって、テーブルがサーバのキャッシュに完全には収まらない、複数のインデックスがある大規模なテーブルに対する処理速度が大幅に早くなりました。Adaptive Server Anywhere の以前のバージョンで使用されていた検証アルゴリズムを使用する場合、-fn オプションを指定できます。

  「検証ユーティリティ (dbvalid)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **[接続] ダイアログでのマルチバイト文字の入力**　Windows のセキュリティ上の慣習に準拠するには、Input Method Editor (IME) を使用して、Sybase Central、Interactive SQL、および Adaptive Server Anywhere コンソール・ユーティリティで使用される [接続] ダイアログの [パスワード] フィールドに日本語や他のアジア諸国のマルチバイト文字を入力することはできません。

  既存のデータベースのパスワードにこれらの文字が含まれる場合、パスワードは、[接続] ダイアログと [ODBC の設定] の両ダイアログの [詳細] タブの [追加接続パラメータ] フィールドに入力できます。ただし、[詳細] タブにパスワードを入力する場合、パスワードは隠されず、プレーン・テキストで表示されるので注意してください。データベースをアップグレードする場合は、マルチバイト文字が含まれないようにパスワードを変更することをおすすめします。

- **DECLARE LOCAL TEMPORARY TABLE 文には所有者名を指定できない**　ソフトウェアの以前のバージョンでは、DECLARE LOCAL TEMPORARY TABLE に所有者名を指定するときに所有者が現在のユーザと異なっていた場合、同じ名前をもつテンポラリ・テーブルを複数作成できました。現在では、所有者名を指定すると、構文エラーが発生します。

- **min_table_size_for_histogram オプションのデフォルト設定の変更**　min_table_size_for_histogram オプションを使用して、ヒストグラムが作成されるテーブルの最小サイズを指定します。このデフォルト値は、100 ローに変更されました。ソフトウェアの以前のバージョンでは、デフォルト値は 1000 ローでした。ソフトウェアの以前のバージョンで作成したデータベースでは、SET OPTION 文を使用してこの設定を変更できます。

- **NULL 定数データ型変換の変更**　以前のバージョンでは、NULL 定数を CHAR、VARCHAR、LONG VARCHAR、BINARY、VARBINARY、または LONG BINARY データ型に変換するときに長さが指定されていない場合、カラムのサイズは 32767 に初期化されていました。現在、このサイズは 0 に初期化されます。

  たとえば、以前、次のクエリは、長さが 32767 として記述されたカラムを返していました。

```
SELECT CAST( NULL AS CHAR )
-- This now returns a CHAR(0) column

SELECT 'abc'
UNION ALL
```

```
SELECT NULL
-- This now returns a CHAR(3) column

SELECT ''
UNION ALL
SELECT NULL
-- This now returns a CHAR(0) column

SELECT IF 1=1 THEN 'abc' ELSE NULL ENDIF
-- This now returns a CHAR(3) column
```

- **ORDER BY 句に序数を使用するときの UPDATE 文とエラー**　序数を使用する ORDER BY 句が含まれる UPDATE 文は、構文エラーを返すようになりました。
- **識別子に関する制限**　識別子には二重円記号または二重引用符を使用できなくなりました。識別子に円記号を使用できるのは、エスケープ文字として使用する場合のみです。

  「識別子」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **EXECUTE IMMEDIATE 文のデフォルト設定の WITH RESULT SET 句**　WITH RESULT SET ON 句を指定すると、EXECUTE IMMEDIATE 文が結果セットを返すことができます。デフォルト設定は WITH RESULT SET OFF です。

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

Mobile Link の動作の変更

ここでは、これまでのバージョンとは異なる動作をリストにして説明します。

- **フックが呼び出される順序の変更**　イベント・フックが呼び出される順序が変更されました。つまり、アップロード・イベントはインクリメンタル更新ごとに繰り返される同期シーケンスの一部にすぎないため、インクリメンタル・アップロードがより効率的になりました。

  「同期イベント・フックの順序」『Mobile Link - クライアント管理』を参照してください。
- **Adaptive Server Anywhere クライアントのパブリケーションのローワイズ分割の修正**　WHERE 句が含まれるパブリケーションは、WHERE 条件を満たすローのみをレプリケートするようになりました。バージョン 8.0.0 ー 9.0.0 には、WHERE 句が不定の値と評価されたときにローをレプリケートするバグがありました。たとえば、パブリケーション WHERE 句に "WHERE val = 1" とあった場合、val が NULL のローもレプリケートされました。このバグは、SQL Remote クライアントと Adaptive Server Anywhere Mobile Link クライアントの両方に影響しました。

Ultra Light の動作の変更

ここでは、これまでのバージョンとは異なる動作をリストにして説明します。

- **Palm OS のステータス管理**　Ebedded SQL または静的型 C++ API が使用された Palm OS アプリケーションの場合、ULPalmExit (ULData::PalmExit) と ULPalmLaunch (ULData::PalmLaunch)

関数は、ステータスや同期情報を管理する必要がなくなり、使用されなくなりました。また、ULData と ULConnection Reopen メソッドも使用されなくなりました。

Palm OS 上のアプリケーションの初期化、接続、終了関数の順序は他のアプリケーションと同じようになりました。ULSetSynchInfo メソッドは、HotSync 同期を制御します。

「Palm アプリケーションに HotSync 同期を追加する」『Ultra Light - C/C++ プログラミング』を参照してください。

- **Palm OS 3.0 のサポート終了**　このリリースでサポートされる最も古いバージョンは、Palm OS 3.5 です。
- **ULEnableGenericSchema 関数 (旧式)**　スキーマのアップグレードが必要な Ultra Light C/C++ アプリケーションは、ULEnableGenericSchema を呼び出す必要がなくなりました。代わりに、ULRegisterSchemaUpgradeObserver 関数を使用します。
- **Ultra Light コンポーネントのテーブル API**　Table オブジェクトの Delete メソッドは、削除後にローを自動的に再表示しなくなりました。以前の動作を維持するには、削除オペレーション後に Relative(0) を使用してローを再フェッチします。
- **Native Ultra Light for Java によるカラム ID とパラメータ ID のキャストは不要**　カラム ID とパラメータ ID を受け入れていたすべてのメソッド、short 型のパラメータを受け入れていた一部のメソッドは、整数を受け入れるよう変更されました。これによって、コードに数値定数をキャストする必要がなくなりました。たとえば、table.getString((short)1); の代わりに、table.getString(1); を使用できるようになりました。

  この変更の結果、9.0.1 ソフトウェアに対応させるために Native Ultra Light for Java アプリケーションは再コンパイルします。ただし、コードの変更は必要ありません。

第 5 章

バージョン 9.0.0 の新機能

目次

バージョン 9.0.0 の新機能

この項では、SQL Anywhere Studio バージョン 9.0.0 のコンポーネントに導入された新機能について説明します。

Adaptive Server Anywhere の新機能

この項では、Adaptive Server Anywhere バージョン 9.0.0 の新機能について説明します。新機能について主要なものからそうでないものまですべてを示し、各機能の詳細についての参照先も記述しています。

新機能のハイライト

- **XML のサポート** Adaptive Server Anywhere 9.0.0 では、XML ドキュメントの保存、リレーショナル・データを XML としてエクスポート、XML のインポート、リレーショナル・データのクエリから XML を返すなど、XML を広範囲にわたってサポートします。
  - **FOR XML 句** RAW、AUTO、EXPLICIT の 3 つのモードで SELECT 文に FOR XML 句を指定できるようになり、XML ドキュメントのクエリ結果を得ることができます。各モードでは、生成される XML の形式をそれぞれ異なるレベルで制御できます。

    「クエリ結果を XML として取得する」『SQL Anywhere サーバ - SQL の使用法』と「SELECT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **for_xml_null_treatment オプション** for_xml_null_treatment オプションを使用すると、FOR XML 句を含むクエリからの NULL 値の返し方を制御できます。

    「for_xml_null_treatment オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
  - **openxml プロシージャ** 「openxml システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
  - **SQL/XML のサポート** SQL/XML は、SQL と XML を組み合わせて使用する場合の仕様を記述した規格草案です。SQL/XML サポートの一部として、Adaptive Server Anywhere は、XML ドキュメントをデータベースに保存できる XML データ型を備えています。

    「XML データ型」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

    Adaptive Server Anywhere では、リレーショナル・データから XML ドキュメントを生成する手段として、FOR XML 句のほかに次の SQL/XML 関数もサポートしています。
    - **XMLAGG 関数** この集合関数は、XML 要素のコレクションから XML 要素のフォレストを生成します。

      「XMLAGG 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
    - **XMLCONCAT 関数** この関数は、渡された XML 値を連結して XML 要素のフォレストを生成します。

「XMLCONCAT 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **XMLELEMENT 関数** この関数は XML 要素を生成します。オプションで、要素の内容、属性、属性の内容を指定できます。

  「XMLELEMENT 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **XMLFOREST 関数** この関数は、XML 要素のフォレストを生成します。

  「XMLFOREST 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **XMLGEN 関数** この関数は、XQuery コンストラクタに基づいて XML 値を生成します。

  「XMLGEN 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **データベース内の HTTP サーバ** Adaptive Server Anywhere データベース・サーバが Web サーバとして動作できるようになりました。このため、Adaptive Server Anywhere データベースと任意の Web ブラウザだけを使用して、Web ベースのアプリケーションを作成し、実行できます。

  この機能によって、データベース・サーバは、標準の SOAP 要求に加えて、標準の HTTP 要求と HTTPS 要求も処理します。使用できるサービス・タイプは、HTTP、HTTPS、XML、RAW、SOAP、DISH です。DISH は SOAP サービスのハンドラです。

  このリリースより前に作成されたデータベースでこの拡張機能を利用するには、アップグレード・ユーティリティを使用してデータベースをアップグレードする必要があります。

  「SQL Anywhere Web サービス」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **インデックス・コンサルタント** インデックス・コンサルタントは、適切なインデックス選択を支援するためのツールです。単一のクエリまたは操作のセットを分析して、データベースに追加するインデックスや削除するインデックスを推奨します。

  このリリースより前に作成されたデータベースでこの拡張機能を利用するには、アップグレード・ユーティリティを使用してデータベースをアップグレードする必要があります。

  「インデックス・コンサルタント」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **64 ビット・バージョンが使用できる** Itanium II チップ搭載の Windows Server 2003 では、ソフトウェアの 64 ビット通常版を使用できます。64 ビットの Linux と HP-UX オペレーティング・システムでは、クライアント／サーバ限定版を使用できます。

SQL の強化

- **選択の前に WITH 句で共通テーブル式を指定できる** 共通テーブル式は、SELECT 文のスコープにのみ存在するテンポラリなビュー定義です。再帰的または非再帰的です。共通テーブル式を使用すると、より簡潔にクエリを記述できる場合があります。また、1 つのクエリの中で複数レベルの集合を実行できます。共通テーブル式を使用できるのは、トップレベルの SELECT

文、ビュー定義内のトップレベルの SELECT 文、または INSERT 文内のトップレベルの文の中だけです。

「共通テーブル式」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **特殊な形式の共通テーブル式を使用して再帰ユニオンを実行できる** 再帰的な共通テーブル式によって、再帰クエリを記述できます。階層的なデータ構造や有向グラフを表すテーブルに対してクエリを実行する場合に、この機能は特に効果的です。どの再帰共通テーブル式も、最初に実行される初期サブクエリと、再帰サブクエリを含みます。再帰サブクエリの FROM 句に指定する必要があるビューへの参照は、前の反復処理でビューに追加されたローを参照します。クエリを実行するデータ構造が何層にもわたる場合は特に、再帰処理を停止する条件の指定に注意してください。

  「再帰共通テーブル式」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **INTERSECT 文と EXCEPT 文のサポート** これらの文では、複数の結果セットの共通部分と差異が計算されます。これは、UNION 文を補うものです。

  詳細については、次の項を参照してください。

  - 「EXCEPT と INTERSECT の使用」『SQL Anywhere サーバ - SQL の使用法』
  - 「集合演算子と NULL」『SQL Anywhere サーバ - SQL の使用法』
  - 「EXCEPT 文」『SQL Anywhere サーバ - SQL リファレンス』
  - 「INTERSECT 文」『SQL Anywhere サーバ - SQL リファレンス』

- **SELECT 文にストアド・プロシージャの結果セットを指定できる** SELECT 文の中で、ベース・テーブルやビューを指定できる位置に、ストアド・プロシージャ・コールを指定できるようになりました。

  ストアド・プロシージャ・コールの統計情報を保存する場合は、アップグレード・ユーティリティを使用してデータベースをアップグレードする必要があります。統計情報がない場合、ストアド・プロシージャ・コールの結果をジョインしようとしても、適切なプランが得られない可能性があります。

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

- **オンライン分析処理機能の追加** 使用可能な SQL 言語に、さまざまな OLAP 機能が追加されました。

  - **ROLLUP 演算** クエリに GROUP BY 句が指定されている場合、ROLLUP 演算によって結果セットに小計のローが追加されます。各小計ローには、GROUP BY の結果セットに含まれるロー・セットの合計が示されます。

    「ROLLUP の使用」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

  - **LIST 関数で順序付きのリストを使用できる** LIST 関数の機能が拡張されて、ソートされた項目のリストを出力できるようになりました。

    「LIST 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **その他の集合関数**　標本ベースと母集団ベースの標準偏差と分散を計算する関数が追加されました。

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

- **CREATE INDEX 文で組み込み関数のインデックスを作成できる**　この機能は、テーブルに新しい計算カラムを追加し、そのカラムのインデックスを作成できる便利な方法です。

  「CREATE INDEX 文」『SQL Anywhere サーバ - SQL リファレンス』と「インデックスの作成」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **すべてのコンテキストで ORDER BY 句を使用できる**　以前のリリースでは、ビュー定義、サブクエリ、または UNION 文内の多くの SELECT 文では ORDER BY 句を使用できませんでしたが、その制限が取り除かれました。

  SELECT 文に ORDER BY 句を指定すると、FIRST 句や TOP 句と一緒に使用する場合などは特に、ビュー定義や集合演算の結果が影響を受けることがあります。それ以外のコンテキストでは、ORDER BY 句を指定しても、オペレーションに違いは生じません。

- **SELECT 文の TOP 句に START AT を指定できる**　START AT を指定すると、結果セットを明示的に制限するクエリの中で、さらに柔軟性を高めることができます。

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

- **制約に名前を指定できる**　検査制約、一意性制約、参照整合性制約に、名前を指定できるようになりました。このため、テーブル全体の制約を変更するのではなく、個々の制約を変更することで、テーブルやカラムの制約を変更できます。

  このリリースより前に作成されたデータベースでこの拡張機能を利用するには、データベースをアンロードして再ロードすることによってそのデータベースのファイル・フォーマットをアップグレードする必要があります。

  「ALTER TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』、「CREATE TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』、「テーブル制約とカラム制約の使い方」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **FROM 句の中でラテラル抽出テーブルによる外部参照が可能**　FROM 句の中で、抽出テーブルとストアド・プロシージャからの外部参照を指定できるようになりました。外部参照の実行を指定するには、LATERAL キーワードを使用します。

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

- **EXECUTE IMMEDIATE 文でエスケープ文字を柔軟に処理できる**　新しい WITH ESCAPES OFF オプションを使用すると、エスケープ文字の処理を抑制できます。この機能によって、ファイル・パスを含む動的な文を簡単に作成できるようになります。

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

- **EXECUTE IMMEDIATE 文が結果セットを返すクエリをサポートする**　この機能によって、ストアド・プロシージャの中で、より動的に文を作成できるようになりました。

「プロシージャでの EXECUTE IMMEDIATE 文の使用」『SQL Anywhere サーバ - SQL の使用法』と「EXECUTE IMMEDIATE 文 [SP]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **CREATE FUNCTION 文と ALTER FUNCTION 文で Transact-SQL 構文を使用できる** Transact-SQL 構文を使用して、呼び出し元の環境へスカラ値を返すユーザ定義関数を定義できるようになりました。

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

- **複数のローの挿入時にオートインクリメント・カラムの値を使用できる** value-sensitive (キーセット駆動型) カーソルを使用してローを挿入する場合、新しく挿入されたローはカーソル結果セットの最後に表示されます。

  このため、直前に挿入されたローのオートインクリメント・カラムの値は、カーソル内の最後の行を選択することによって確認できます。たとえば、Embedded SQL の場合、FETCH ABSOLUTE -1 *cursor-name* を使用することで値を取得できます。

  「カーソルによるローの変更」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **リモート・データ・アクセスで UUID/GUID カラムを処理できる** リモート・データ・アクセスで、Microsoft SQL Server のユニーク識別子カラムを管理できるようになりました。

  「データ型変換：Microsoft SQL Server」『SQL Anywhere サーバ - SQL の使用法』と「UNIQUEIDENTIFIERSTR データ型」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **リモート・データ・アクセスでリモート接続に名前を指定できる** ODBC を介してのリモート・データ・アクセス接続に名前を指定できるようになりました。このため、特定の接続を切断することができます。

  「ODBC を使用したリモート・データ・アクセスの接続の管理」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **式のデータ型を返す新しい関数の追加** EXPRTYPE 関数は、式のデータ型を返します。

  「EXPRTYPE 関数 [その他]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。.

- **EXIT 文の強化** Interactive SQL の EXIT 文は、Interactive SQL に終了コードを設定できるようになりました。

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

- **OUTPUT 文に ASIS キーワードを指定できる** ASIS を指定すると、エスケープ処理をせずに値がそのままファイルに書き込まれます。

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

- **インデックスと外部キーを変更できる**　ALTER INDEX 文を使用すると、インデックスと外部キーの名前を変更できます。また、プライマリまたは外部キー・インデックスに加えて、ユーザ作成インデックスについても、インデックスのタイプをクラスタードまたはノンクラスタードに変更できます。

  このリリースより前に作成されたデータベースでクラスタード・インデックスを利用するには、データベースをアンロードして再ロードすることによってそのデータベースのファイル・フォーマットをアップグレードする必要があります。

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

- **クエリの中で異なる集合を複数使用できる**　集合関数の引数として DISTINCT *column-name* を指定できます。ソフトウェアの以前のバージョンでは、DISTINCT 引数を伴う集合関数は、クエリに 1 つだけ指定できました。現在のバージョンでは、複数の集合関数を使用できます。以前のバージョンでは次のクエリを使用できませんでしたが、バージョン 9 では有効です。

  ```
  SELECT count( DISTINCT first_name ),
         count( DISTINCT last_name )
  FROM contact
  ```

- **サポートされるすべての言語のイベント・スケジュールで、英語のフルスペルの曜日名と省略形の曜日名が認識される**　イベントを作成する場合、データベース・サーバでは、Adaptive Server Anywhere がサポートしているすべての言語において、英語のフルスペルの曜日名と省略形の曜日名の両方が認識されます。以前のバージョンでは、英語以外の言語のスケジュールでは、フルスペルの英語の曜日名を指定する必要がありました。

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

- **プロシージャ・テキストを隠してロジックの機密を保持**　SET HIDDEN オプションを使用して、ストアド・プロシージャ、関数、トリガ、ビューに含まれるロジックを隠すことができます。これによって、ストアド・プロシージャ、関数、トリガ、ビュー内のロジックを明らかにせずに、アプリケーションやデータベースを配布できます。

  このリリースより前に作成されたデータベースでこの拡張機能を利用するには、アップグレード・ユーティリティを使用してデータベースをアップグレードする必要があります。

  「プロシージャ、関数、トリガ、ビューの内容を隠す」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

管理とスケーラビリティの強化

- **より詳細なリターン・コードを返す検証ユーティリティ**　検証ユーティリティ (dbvalid) は、異常の発生原因を示す、より詳細なリターン・コードを返します。

  「検証ユーティリティ (dbvalid)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **2 つの新しいサーバ・プロパティ**　2 つの新しいサーバ・プロパティが追加されました。CommandLine は、サーバを起動するときに使用したコマンド・ラインを表示し、CompactPlatformVer は、PlatformVer サーバ・プロパティの縮小バージョンを表示します。

  「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しい sp_remote_primary_keys ストアド・プロシージャ**　リモート・データ・アクセスを使用するリモート・テーブルに関するプライマリ・キー情報を取得するために、sp_remote_primary_keys というストアド・プロシージャが追加されました。

  このリリースより前に作成されたデータベースでこの拡張機能を利用するには、アップグレード・ユーティリティを使用してデータベースをアップグレードする必要があります。

  「sp_remote_primary_keys システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **接続の通信リンク名を返す新しい connection_property**　CommNetworkLink という新しい接続プロパティが、接続の通信リンクの名前を返します。

  「接続レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **NetWare が完全な文字セット変換をサポート**　バージョン 8.x の NetWare では、シングルバイト対シングルバイトの変換がサポートされていました。バージョン 9.0 では、他のプラットフォームでサポートされるすべての文字セットが、NetWare でもサポートされます。

- **アンロード・ユーティリティでカラム・リストをアンロードできる**　アンロード・ユーティリティ (dbunload) で、reload.sql ファイルに生成される、LOAD TABLE 文のカラム・リストをアンロードできるようになりました。これによって、テーブル内のカラムの順序を簡単に変更できます。

  「アンロード・ユーティリティ (dbunload)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **LDAP によるデータベース・サーバの登録**　データベース・サーバを LDAP サーバに登録できるようになったので、クライアントと検索ユーティリティ (dblocate) は、LDAP サーバに対してデータベース・サーバの検索を実行できます。このため、WAN やファイアウォールを経由して動作しているクライアントは、IP アドレスを指定することなくデータベース・サーバを検索できます。LDAP は TCP/IP でネットワーク・サーバ上でのみ使用されます。

  「LDAP サーバを使用した接続」『SQL Anywhere サーバ - データベース管理』または「LDAP プロトコル・オプション [LDAP]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **接続数が多い場合の処理の向上**　200 を超える接続がある場合、大量の接続を適切に処理するために、活性タイムアウト値が自動的に増加します。

  「-tl サーバ・オプション」『SQL Anywhere サーバ - データベース管理』と「LivenessTimeout 接続パラメータ [LTO]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **要求ログのフィルタリングとホスト変数のサポート**　要求ログへの出力をフィルタリングできるようになり、特定の接続や特定のデータベースの要求だけを入れることができます。また、ホスト変数値を要求ログに出力できるようになりました。

  「sa_server_option システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』、「パフォーマンスのモニタリングと改善」『SQL Anywhere サーバ - SQL の使用法』、「sa_get_request_times システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレン

ス』、「-zr サーバ・オプション」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **BACKUP 文と dbbackup でログ・コピーの名前を変更できる** BACKUP 文とバックアップ・ユーティリティ (dbbackup) を使用して、ログ・コピーの名前を変更できます。

  「バックアップ・ユーティリティ (dbbackup)」 『SQL Anywhere サーバ - データベース管理』と「BACKUP 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **START DATABASE 文でチェックポイントでのログ・トランケーションと読み込み専用モードを指定できる** START DATABASE 文を使用すると、有効なチェックポイントでのログ・トランケーション、または読み込み専用モードを指定して、データベースを起動できます。

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

- **Adaptive Server Anywhere における異なる監査オプションのサポート** Adaptive Server Anywhere の以前のバージョンでは、監査のオンとオフを選択できました。現在のバージョンでは、監査するオプションを指定できるようになりました。

  「sa_disable_auditing_type システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』または「sa_enable_auditing_type システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **event_parameter 関数に渡せる 3 つの新しい値** 3 つの新しい値を event_parameter 関数に渡すことができます。ScheduleName は、イベントを発生したスケジュールの名前を返します。AppInfo は、イベントを引き起こした接続の connection_property ('AppInfo') の値を返します。DisconnectReason は、接続が終了した原因を示す文字列を返します。

  「EVENT_PARAMETER 関数 [システム]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **ネットワーク・サーバに同時に接続しているユーザ数を示す新しいサーバ・プロパティ** 新しい LicensesInUse プロパティによって、ネットワーク・サーバに現在同時に接続しているユーザ数が判断されます。各同時接続ユーザは、接続数ではなく、サーバに接続しているユニークなクライアント・ネットワーク・アドレスの数で判断されます。たとえば、3 台のクライアント・コンピュータがサーバに接続していて、各クライアント・コンピュータが 2 つの接続を持つ場合、プロパティ ('LicensesInUse') を選択すると "3" が返されます。

  詳細については、「サーバ・レベルのプロパティ」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **サービス作成 [dbsvc] ユーティリティでサービスの起動と停止が可能** サービス作成 [dbsvc] ユーティリティに、2 つの新しいオプションが追加されました。1 つは **Dbsvc -u** *service_name* で、service_name で指定されるサービスを起動します。もう 1 つは *dbsvc -x service_name* で、service_name で指定されるサービスを停止します。

  「Windows 用サービス・ユーティリティ (dbsvc)」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **LocalOnly プロトコル・オプション [LOCAL] をサポートするネットワーク・サーバ** サーバで、LocalOnly プロトコル・オプション [LOCAL] を使用できます。LocalOnly プロトコル・オ

プションを YES に設定してサーバを実行すると、ネットワーク・サーバは、接続や CPU の制限を受けないパーソナル・サーバとして動作します。

「LocalOnly プロトコル・オプション [LOCAL]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Address Windowing Extensions 使用時の最小データベース・サーバ・キャッシュ・サイズの変更** Windows 2000、Windows XP、Windows Server 2003 で Address Windowing Extensions (AWE) を使用する場合の、データベース・サーバ・キャッシュの最小サイズが 2 MB になりました。以前のリリースにおける AWE 使用時の最小キャッシュ・サイズは 3 GB ー 256 MB でした。

  「-cw サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **ドライブ・タイプを指定する新しいデータベース・プロパティ** データベース・ファイルがあるドライブについての情報を提供する、新しい DriveType データベース・プロパティが追加されました。

  「データベース・プロパティの概要」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Adaptive Server Anywhere NetWare の高速化** NetWare 版の Adaptive Server Anywhere サーバは、CLIB ではなく LibC を使用するようになりました。LibC は C ランタイム・ライブラリで、これまでの CLIB ライブラリよりも、NetWare オペレーティング・システムの新しいカーネルとの相互関係に優れています。NetWare 版のすべてのクライアント側ソフトウェア (dblib、dbisql、dbconsole、dbremote など) は、引き続き CLIB を使用します。これには NetWare の最大ファイル・サイズを NTFS と同じサイズに増加できるという利点があります。複数の CPU を使用し (ある場合)、TCP と SPX が Winsock を使用することで、以前のバージョンよりも高速になります。

  「Adaptive Server Anywhere の制限」『SQL Anywhere サーバ - データベース管理』と「バージョン 9.0 での動作の変更」 224 ページを参照してください。

- **NetWare における外部関数の強化** NetWare の外部関数または外部ストアド・プロシージャは、名前が競合することなく複数の NLM を使用できるようになりました。

  詳細については、「外部関数のプロトタイプ」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **エラー・メッセージの言語を接続ごとに指定可能** データベース・サーバへの各接続において、データベース・サーバから出力されるエラー・メッセージやさまざまな文字列の言語を要求できるようになりました。接続が使用する言語は、サーバが使用する言語とは別です。データベース・サーバでは、接続が要求する言語を使用して日付文字列を解釈することもできます。

- **プロセッサ・タイプを識別する 2 つの新しいサーバ・プロパティの追加** 2 つの新しいサーバレベル・プロパティが追加されました。1 つは ProcessorArchitecture であり、プロセッサ・タイプを識別します。もう 1 つは NativeProcessorArchitecture であり、プロセッサがエミュレートされるプラットフォーム上で、ネイティブのプロセッサ・タイプを識別します。

「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **データベースの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は無関係**　CREATE DATABASE 文、初期化 [dbinit] ユーティリティ、および [データベース作成] ウィザードを使用して、パスワードの大文字と小文字を区別するかどうかを指定します。パスワードの大文字と小文字を区別する設定は、文字列の比較に使用されるデータベースの大文字と小文字を区別する設定とは関係ありません。新しい CaseSensitivePasswords データベース・プロパティを使用して、データベースのパスワードの大文字と小文字を区別する設定を確認できます。

  「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』と「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバ - データベース管理』を参照してください。

パフォーマンスの強化 (クエリの最適化)

次に示す新機能はクエリの最適化を強化する機能ですが、これらの機能を使用するのにユーザの操作は必要ありません。これらの機能は、ユーザが介入しなくても実行されます。クエリ実行プランを調べると、このような最適化の効果を確認できます。

最適化の強化機能を使用するためにデータベースをアップグレードする必要はありませんが、バージョン 9 ソフトウェアで作成されたデータベースが最も効率的に処理されます。

- **コストベースのサブクエリの最適化**　オプティマイザは、サブクエリが使用できる最適化のスコープを大幅に拡張しました。以前のリリースでは、サブクエリは、セマンティックなクエリの最適化の中でジョインとして記述し直すか、クエリの残りの部分とは別に最適化されていました。現在のバージョンでは、ジョインとして記述し直すには複雑すぎるサブクエリも、クエリの一部としてそのまま最適化できるようになりました。

- **バッファされたローをフェッチすることで逐次スキャンのパフォーマンスを向上**　逐次テーブル・スキャンでデータベース・ページからローを読み込む場合、ローをバッファにコピーしてから、ユーザにローを返します。クエリの複雑さによりますが、これによって時間が大幅に短縮されます。

- **効率的に実行される TOP N クエリ**　TOP N 句を使用するクエリの実行アルゴリズムが新しくなり、実行速度が速くなりました。

  「上位 N のソート・アルゴリズム」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **ヒストグラムに保持する頻度を決定する新しいアルゴリズム**　以前のバージョンでは、カラム・ヒストグラムは、選択性が 1% を超える値に対して単一バケットを作成していました。現在のバージョンでは単一バケットの条件が緩められ、代わって、ヒストグラムは単一バケットの最小数を保持しようとします。

  「オプティマイザの推定とカラム統計」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **現在キャッシュされているクエリ・プラン数を表示する QueryCachedPlans プロパティ**　新しい QueryCachedPlans プロパティを使用すると、特定の接続やすべての接続において、現在キャッシュされているクエリ実行プランの数を表示できます。QueryCachePages、

QueryOptimized、QueryBypassed、QueryReused と組み合わせて使用することで、max_plans_cached オプションの最適な設定を判断できます。

「接続レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』と「データベース・プロパティの概要」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **プロシージャの文でプランが高速にキャッシュされる**　アクセス・プランがキャッシュされる文のスコープが拡張されて、プロシージャによって結果セットが呼び出し元の環境に返される、ストアド・プロシージャ内のクエリも含まれるようになりました。この強化機能によって、いくつかの文は再び最適化しなくても済みます。

  「プランのキャッシュ」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **各インデックスが更新されるたびにインデックスの統計情報が維持される**　各インデックスが更新されるたびに、カタログ・テーブルのインデックスを含めてすべてのインデックスの統計情報が維持され、実質的にパフォーマンスを損なうことなくオプティマイザに対して正確な統計情報を提供します。統計は、インデックスの各統計に対して 1 つのローという形式で SYSATTRIBUTE で維持されます。

  このリリースより前に作成されたデータベースでこの拡張機能を利用するには、データベースをアンロードして再ロードすることによってそのデータベースのファイル・フォーマットをアップグレードする必要があります。

  「ISYSATTRIBUTE システム・テーブル」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

パフォーマンスの強化 (サーバ・オペレーション)

- **パフォーマンス・モニタ統計値の追加**　[通信 :使用中のライセンス]、[接続数] という 2 つの新しいパフォーマンス・モニタ統計値が追加され、使用中の接続数を追跡できるようになりました。

  「通信の統計値」『SQL Anywhere サーバ - SQL の使用法』と「その他の統計値」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **UNLOAD 文と UNLOAD TABLE 文への APPEND { ON | OFF } オプションの追加**　新しい APPEND オプションを使用すると、アンロードしたデータを指定のファイルの最後に追加できます。

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

- **テンポラリ・テーブルを NOT TRANSACTIONAL として宣言できる**　NOT TRANSACTIONAL を使用すると、テーブルは COMMIT や ROLLBACK の影響を受けません。テーブルにアクセスするプロシージャが COMMIT なしで繰り返し呼び出される場合に、この拡張機能は便利です。

  「CREATE TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』と「DECLARE LOCAL TEMPORARY TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **継続的なインデックス統計情報**　候補となるインデックスの物理的プロパティに関して正確な統計情報を管理することにより、オプティマイザはコストに基づいて、使用するインデック

スを容易に選択できます。統計情報は SYSATTRIBUTE にあり、インデックスが更新されるたびに保持されます。また、VALIDATE 文は、指定のインデックスの統計情報が正確であるかどうかを検証し、正確でない場合はエラーを生成します。これによって、実質上パフォーマンス・コストをかけることなく、正確な統計がオプティマイザに提供されます。

このリリースより前に作成されたデータベースでこの拡張機能を利用するには、データベースをアンロードして再ロードすることによってそのデータベースのファイル・フォーマットをアップグレードする必要があります。

「ISYSATTRIBUTE システム・テーブル」『SQL Anywhere サーバ - SQL リファレンス』と「VALIDATE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **ptimistic_wait_for_commit オプションの追加** このオプションは、トランザクションがプライマリ・ローの前に外部ローを追加した場合に、バージョン 5.x のロック動作を模倣するためのものです。汎用的なオプションではありませんが、バージョン 5.x のアプリケーションをバージョン 8.x 以降に移行する場合に役立ちます。

  「wait_for_commit オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **拡張プロパティ関数の追加** 新しい DB_EXTENDED_PROPERTY 関数は、プロパティ固有の文字列パラメータをオプションで指定できる点を除いて、DB_PROPERTY 関数に似ています。

  「DB_EXTENDED_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **2 つの新しいプロパティの追加** FileSize と FreePages という 2 つの新しいプロパティが追加されました。各プロパティとも、オプションの引数で、プロパティが要求される DB 領域を指定できます。

  「データベース・プロパティの概要」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **サーバのクワイエット・モードの強化** サーバのクワイエット・モードとエラー・ログのオプションが強化され、さまざまなメッセージの出力を抑制できるようになりました。また、-q オプションは -qw オプションに変更され、-Q オプションは -qi オプションに変更されました。

開発ツールと管理ツール

- **Adaptive Server Anywhere プラグインの変更** Sybase Central の Adaptive Server Anywhere プラグインの構成が変更されました。これまでプロパティ・シート、ダイアログ・ボックス、左ウィンドウ枠内のフォルダに表示されていた情報の多くが、右ウィンドウ枠内のタブに表示されるようになりました。たとえば、外部キーの情報を表示する場合、現在のバージョンでは、左ウィンドウ枠で外部キーを持つテーブルを選択し、右ウィンドウ枠で [外部キー] タブを選択します。以前のバージョンでは、左ウィンドウ枠に [外部キー] フォルダがありました。

  このほかにも、プラグインには次に示すようないくつかの変更が加えられています。

  - テーブル・エディタは別のウィンドウではありません。テーブルは、Sybase Central の右ウィンドウ枠で直接編集します。

- 一度に複数のウィンドウを開きたい場合、Sybase Central の右ウィンドウ枠または別のコード・エディタのウィンドウでストアド・プロシージャ、関数、トリガ、およびイベントを編集できます。
- ツールバーのボタンに、選択されているオブジェクト固有のオプションが含まれるようになりました。
- SQL 文のログとサーバ・メッセージは、Sybase Central のメイン・ウィンドウに直接表示できます (サーバ・メッセージ・ウィンドウに同じ情報が表示されます)。この情報を表示するには、Sybase Central で、[ファイル] – [サーバ・メッセージと実行された SQL] を選択します。Sybase Central のメイン・ウィンドウの下部に、[サーバ・メッセージと実行された SQL] ウィンドウ枠が表示されます。
- Adaptive Server Anywhere プラグインにはさまざまな新しいウィザードが追加されており、その指示に従うことで、テーブル、ユニークな制約、Web サービスを作成できます。

- **Adaptive Server Anywhere プラグインにおけるクリップボード・サポートの強化** Adaptive Server Anywhere プラグインにおけるクリップボード・サポートが強化され、Sybase Central 内の大半のオブジェクトを、Interactive SQL やテキスト・エディタなど、他のアプリケーションにコピーして貼り付けることができます。オブジェクトを他のアプリケーションにコピーする場合は、選択するオブジェクトに応じて、オブジェクトの名前または SQL が表示されます。たとえば、Sybase Central の中でインデックスをコピーし、それをテキスト・エディタに貼り付ける場合、そのインデックスの CREATE INDEX 文が表示されます。

  「SQL Anywhere プラグインのデータベース・オブジェクトのコピー」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **デバッガの変更** ストアド・プロシージャと Java クラスの両方をデバッグできるデバッガが、Sybase Central に統合されました。ユーザ・インタフェースは再設計されています。

  「プロシージャ、関数、トリガ、イベントのデバッグ」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **Adaptive Server Anywhere Console ユーティリティの強化** インタフェースの変更、複数接続、ソート、ドラッグ・アンド・ドロップのサポートなど、Adaptive Server Anywhere コンソール・ユーティリティが多くの点で強化されました。

- **Sybase Central と Interactive SQL の高速起動** Windows 上で Sybase Central または Interactive SQL を起動するときの、アプリケーションの起動時間の短縮を目的とする高速ランチャが、Sybase Central と Interactive SQL に組み込まれています。Adaptive Server Anywhere 9.0.0 を実行すると、2 つのバックグラウンド・プロセスが開始します。これは *dbisqlg.exe* のインスタンスと *scjview.exe* のインスタンスであり、Interactive SQL と Sybase Central の高速ランチャ・プロセスにそれぞれ相当します。これらの実行プログラムは両方とも、ユーザーがログインすると起動します。

  「高速ランチャの使用」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Interactive SQL の構文を強調表示するエディタ** Interactive SQL の [オプション] ダイアログを使用すると、Interactive SQL の [SQL 文] ウィンドウ枠に入力される構文をどのように表示するかを設定できます。

「[オプション] ダイアログ : [エディタ] タブ」 『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

- **Interactive SQL からの印刷** Interactive SQL の [SQL 文] ウィンドウ枠やグラフィカル・プランの内容を印刷できます。

  「Interactive SQL メイン・ウィンドウの説明」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **グラフィカルなプランの強化** グラフィカルなクエリ・アクセス・プランの表示が、多くの点で強化されました。
  - あるオペレータから別のオペレータへ渡されるローの数は、線の太さを変えることで表されます。
  - 低速なオペレーションは、赤色の枠線で強調表示されます。
  - 統計情報の表示が拡張され、再編成されています。
  - アクセス・プランを印刷できます。
- **データベース・ユーティリティで @filename パラメータを使用できる** Interactive SQL (dbisql)、言語選択ユーティリティ (dblang)、Adaptive Server Anywhere コンソール・ユーティリティ (dbconsole) を除くすべてのデータベース管理ユーティリティでは、@file 構文を使用して、ファイルに含まれるパラメータを指定できるようになりました。ファイル名は設定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。複数のファイルの指定も可能であり、ファイル指定子をコマンド・ライン・スイッチと一緒に使用することもできます。なお、@file 構文は再帰的でないことに注意してください。

  「@data サーバ・オプション」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Interactive SQL で結果の横にロー番号を表示できる** Interactive SQL には、結果の横にロー番号を表示するオプションがあります。このオプションは、Interactive SQL の [オプション] ダイアログの [結果] タブで設定します。

  「[オプション] ダイアログ : [結果] タブ」 『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

- **Interactive SQL を .SQL ファイルのデフォルト・エディタとして設定できる** Windows プラットフォームにおいて、*.SQL* ファイルをダブルクリックすると Interactive SQL がファイルを開くように、ファイルの関連付けを作成できます。

  「Interactive SQL ユーティリティ (dbisql)」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Interactive SQL の [コマンド履歴] ダイアログの強化** Interactive SQL の [コマンド履歴] ダイアログでは、ウィンドウで複数のコマンドを選択できることに加えて、コマンドをコピーしたり削除したりできます。コマンド履歴は、Interactive SQL セッション間で保持されます。

  「SQL 文の印刷」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **警告メッセージの W プレフィクス** 9.0 より以前のバージョンでは、警告メッセージとエラー・メッセージのプレフィクスはすべて、I または E でした。バージョン 9.0 では、警告メッ

セージのプレフィクスは W です。この変更は、dbmlsrv9、dbmlsync、dbremote、ssremote、dbltm、ssqueue に影響します。

Mobile Link の新機能

次に、バージョン 9.0.0 で導入したソフトウェアに加えられた変更と追加を示します。

- **サーバ起動同期**　サーバ起動同期の機能を使用すると、統合データベースから Mobile Link 同期を起動できます。つまり、データ更新をリモート・データベースに送信できます。Mobile Link コンポーネント (Notifier) は、同期を起動する統合データベース内の変更の種類と、更新されたメッセージを受信するリモートの選択方法を決定するための、プログラム可能なオプションを備えています。リモート・コンポーネント (リスナ) では、リモートの応答方法を決定します。

  「サーバ起動同期の概要」『Mobile Link - サーバ起動同期』を参照してください。

- **ファイルベースのダウンロード**　ダウンロードをファイルとして処理できるようになりました。このため、電子メール、ftp、ディスク、マルチキャストによるファイル配布など、ファイルを任意の方法で配布できます。このリリースの場合、この機能を使用できるのは Adaptive Server Anywhere リモート・データベースだけです。

  「Mobile Link ファイルベースのダウンロード」『Mobile Link - サーバ管理』を参照してください。

Moblie Link サーバの強化

- **接続スクリプト begin_publication と end_publication の追加**　2 つの新しいスクリプトが追加されました。これらが使用されるケースの 1 つは、ファイルベースのダウンロードの実装です。

  「begin_publication 接続イベント」『Mobile Link - サーバ管理』と「end_publication 接続イベント」『Mobile Link - サーバ管理』を参照してください。

- **接続スクリプト authenticate_parameters の追加**　カスタム認証を行うための新しいスクリプトが追加されました。認証の中でこの新しいスクリプトが起動されてから、begin_synchronization スクリプトが起動されます。

  「authenticate_parameters 接続イベント」『Mobile Link - サーバ管理』を参照してください。

- **文字列に埋め込まれたブランクを削除するオプションの追加**　dbmlsrv9 -b オプションを指定すると、同期の際、VARCHAR または LONG VARCHAR データ型のカラムの文字列から、後続ブランクが削除されます。

  「-b オプション」『Mobile Link - サーバ管理』を参照してください。

- **.old 拡張子を持つ新しいログ・ファイルを開始するオプション**　dbmlsrv9 -on オプションを指定すると、Mobile Link サーバ・ログが使用するディスク領域について、厳しい制限を課すことができます。

  「-on オプション」『Mobile Link - サーバ管理』を参照してください。

- **ログの進行状況オフセット**　Mobile Link サーバでは、進行状況のオフセット、最終アップロード時間、最終ダウンロード時間をレポートできます。この情報を入手するには、dbmlsrv9 の -vp オプションまたは -v+ オプションを使用します。

  「-v オプション」『Mobile Link - サーバ管理』を参照してください。

- **.NET と Java 同期論理におけるエラーと警告の処理**　Mobile Link サーバでエラーと警告を処理するための論理を追加できます。

  「Java での Mobile Link サーバ・エラーの処理」『Mobile Link - サーバ管理』と「.NET での Mobile Link サーバ・エラーの処理」『Mobile Link - サーバ管理』を参照してください。

- **Mobile Link システム・テーブルへの追加**　ml_user テーブルと ml_subscription テーブルに、last_upload_time と last_download_time という 2 つの新しいカラムが追加されました。デフォルトは NOT NULL で、デフォルト時刻は 1900 年 1 月 1 日 00:00:00 です。

  また、ml_subscription には、subscription_id カラムも追加されました。publication_name カラムには、パブリケーション名が格納されます。

  「ml_user」『Mobile Link - サーバ管理』と「ml_subscription」『Mobile Link - サーバ管理』を参照してください。

Adaptive Server Anywhere クライアントの強化

- **アップロードのみの同期処理**　アップロードのみの同期処理を選択できるようになりました。

  dbmlsync に関する「-uo オプション」『Mobile Link - クライアント管理』を参照してください。

- **ダウンロードのみの同期処理**　ダウンロードのみの同期処理を選択できるようになりました。

  「-ds オプション」『Mobile Link - クライアント管理』と「DownloadOnly (ds) 拡張オプション」『Mobile Link - クライアント管理』を参照してください。

- **ウィンドウ・メッセージによる同期の起動**　ウィンドウ・メッセージを dbas_synchronize として登録し、これを dbmlsync のトップレベル・ウィンドに送信することで、dbmlsync を起動し、同期処理を実行できます。

- **起動時に dll をロード (Windows CE)**　新しい dbmlsync -pd オプションを使用すると、起動時にロードする DLL を指定できます。Windows CE 上で dbmlsync を使用する場合は、このオプションを使用してください。

  「-pd オプション」『Mobile Link - クライアント管理』を参照してください。

- **スキーマをアップグレードまたは変更する手段の追加**　dbmlsync の -i オプションと SiteScriptName (sn) 拡張オプションに代わって、sp_hook_dbmlsync_schema_upgrade フック・ストアド・プロシージャが追加されました。

  「sp_hook_dbmlsync_schema_upgrade」『Mobile Link - クライアント管理』を参照してください。

- **Mobile Link の終了コード**　dbmlsync セッションの中で複数の同期処理を実行している場合は特に、その同期処理の成功や失敗を追跡し、記録することを支援するために、sp_hook_dbmlsync_process_exit_code という新しいクライアント・イベント・フック・プロシー

ジャが追加されました。また、sp_hook_dbmlsync_abort フックの #hook_dict テーブルには、新しい値 (終了コード) が設定されます。

「sp_hook_dbmlsync_process_exit_code」『Mobile Link - クライアント管理』と「sp_hook_dbmlsync_abort」『Mobile Link - クライアント管理』を参照してください。

- **スケジューリングの強化** スケジュールを指定する場合、新しい HoverRescanThreshold (hrt) 拡張オプションまたは sp_hook_dbmlsync_log_rescan フックを使用することで、ログをスキャンする時間を短縮できます。

  「HoverRescanThreshold (hrt) 拡張オプション」『Mobile Link - クライアント管理』と「sp_hook_dbmlsync_log_rescan」『Mobile Link - クライアント管理』を参照してください。

  英語以外の言語でも、スケジュールの中で省略形の英語の曜日名を使用できるようになりました。以前のバージョンでは、英語以外の言語のスケジュールでは、フルスペルの英語の曜日名を指定する必要がありました。

  スケジュール構文に、2 つの新しいキーワードが追加されました。キーワードの 1 つは **INFINITE** です。これは、dbmlsync に対して、次の同期が通知されるまでいつまでも待つことを指定します。もう 1 つのキーワードは **0** です。日付としての 0 はその月の最後の日を指定します。

  「Schedule (sch) 拡張オプション」『Mobile Link - クライアント管理』を参照してください。

Ultra Light クライアントの強化

- **HotSync conduit のトラブルシューティング支援の強化** HotSync ログにトラブルシューティング情報を記録するように、HotSync conduit を設定できます。

パフォーマンスとモニタリングの強化

- **スキーマの変更がない場合の dbmlsync のパフォーマンス向上** dbmlsync はデフォルトで、各同期の前にスキーマ情報をロードしなくなりました。これによって、低速なハンドヘルド・デバイスにおける同期が、一般に 20 秒早くなります。

  「-sc オプション」『Mobile Link - クライアント管理』を参照してください。

- **Windows CE 上での dbmlsync のパフォーマンスの向上** dbmlsync は、Windows CE 上で dbtool9.dll を使用しなくなりました。つまり、メモリの使用量が少なくなります。

- **Mobile Link モニタのコマンド・ライン・オプション** Mobile Link モニタは、さまざまなオプションを指定して、コマンド・ラインから起動できるようになりました。

  「Mobile Link モニタの起動」『Mobile Link - サーバ管理』を参照してください。

- **リダイレクタの強化** 新しく追加された LOG_LEVEL パラメータで、冗長性レベルを制御できます。

  「リダイレクタのプロパティの設定 (サーバ・グループをサポートするリダイレクタの場合)」『Mobile Link - サーバ管理』を参照してください。

- **活性の向上** TCP/IP を介して接続している場合、切断された接続がこれまでより早く検出されるようになりました。接続が切断されたときに Mobile Link ワーカ・スレッドが迅速に解放されるため、スループットが向上します。

その他

- **警告メッセージの W プレフィクス**　9.0 より以前のバージョンでは、警告メッセージとエラー・メッセージのプレフィクスはすべて、I または E でした。バージョン 9.0 では、警告メッセージのプレフィクスは W です。この変更は、dbmlsrv9、dbmlsync、dbremote、ssremote、dbltm、ssqueue に影響します。

SQL Remote の新機能

SQL Remote バージョン 9.0.0 に、次の新機能が追加されています。

- **警告メッセージの W プレフィクス**　9.0 より以前のバージョンでは、警告メッセージとエラー・メッセージのプレフィクスはすべて、I または E でした。バージョン 9.0 では、警告メッセージのプレフィクスは W です。この変更は、dbmlsrv9、dbmlsync、dbremote、ssremote、dbltm、ssqueue に影響します。

Ultra Light の新機能

Ultra Light 開発は、2 種類のプログラミング・インタフェースを使用して行うことができます。

- **Ultra Light コンポーネント**　Ultra Light コンポーネントは、アプリケーション短期開発ツールのユーザに、Ultra Light データベースと同期の機能をもたらします。これらのコンポーネントには、サポートされている開発ツールごとに使い慣れたインタフェースが用意されています。Ultra Light コンポーネントは、簡単なテーブル・ベースのデータ・アクセス・インタフェースのほか、より複雑なクエリ用として動的 SQL も備えています。

  Ultra Light コンポーネントは、バージョン 8.0.2 で導入されました。

- **静的開発モデル**　Embedded SQL、静的型 C++ API、静的型 Java API は引き続き使用可能です。マニュアルの中では、コンポーネントと区別するために静的インタフェースとして参照されています。

特に次の点に注意してください。

- 「**Native Ultra Light for Java**」は、C/C++ Ultra Light Runtime を使用する Ultra Light コンポーネントです。Ultra Light の「**静的型 Java API**」は、以前のリリースで使用可能な pure Java ソリューションであり、コンパイル時にクエリを指定する必要があります。
- 「**Ultra Light for C++**」は、コンポーネント・インタフェースです。Ultra Light の「**静的型 C++ API**」は、以前のリリースで使用可能な静的インタフェースであり、コンパイル時にクエリを指定する必要があります。
- 「**Embedded SQL**」は静的インタフェースであり、コンパイル時にクエリを指定する必要があります。

次に、バージョン 9.0 で導入したソフトウェアに加えられた変更と追加を示します。

- **新しいコンポーネント**　AppForge MobileVB、eMbedded Visual Basic、Java のコンポーネントに加えて、次のコンポーネントが導入されました。
  - **Ultra Light .NET**　Visual Studio .NET 環境を使用する開発用のコンポーネント。このコンポーネントで構築したアプリケーションは、.NET Compact Framework (バージョン 1.05.0000 以上) をサポートするデバイスに配備できます。

    「Ultra Light.NET の概要」『Ultra Light - .NET プログラミング』を参照してください。
  - **C++ コンポーネント**　C++ コンパイラを使用する開発用のコンポーネント。

    「C/C++ 開発者用 Ultra Light の概要」『Ultra Light - C/C++ プログラミング』を参照してください。
- **Pocket IE のサポート**　eMbedded Visual Basic コンポーネントが ActiveX コンポーネントにアップグレードされました。JScript を使用する開発、つまり、Windows CE デバイス上の Pocket IE から実行するアプリケーションに対するサポートが追加されました。
- **動的 SQL**　Ultra Light コンポーネントでは、バージョン 8.0.2 で提供されていたテーブルベースのデータ・アクセス・インタフェースに加えて、マルチテーブル・ジョインなど、より複雑なクエリに対して動的 SQL を使用できるようになりました。
- **接続パラメータ**　Ultra Light コンポーネント (C++ を除く) の接続パラメータは、単一の文字列ではなく個々のプロパティとして公開されています。このため、接続の問題をデバッグしやすくなり、より簡単に接続を管理できるようになりました。

  「Ultra Light の接続文字列パラメータのリファレンス」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **MobileVB コンポーネントのドラッグ・アンド・ドロップ**　MobileVB コンポーネントをフォームにドラッグできるようになりました。コンポーネントのプロパティは、コードと同様、設計環境で設定できます。

  「Ultra Light for AppForge のアーキテクチャ」『Ultra Light - AppForge プログラミング』を参照してください。
- **マルチプロセス・アクセス**　C++ コンポーネントは、複数のプロセスからのアクセスをサポートします。このモデルを使用するアプリケーションを開発するには、個々の Ultra Light データベース・エンジンとアプリケーションを、別の Ultra Light ランタイム・ライブラリにリンクする必要があります。

  「アプリケーションのコンパイルとリンク」『Ultra Light - C/C++ プログラミング』を参照してください。
- **同時同期処理**　以前のリリースでは、同期中のデータへのフル・アクセスは禁止されていました。現在のリリースでは、同期のダウンロード・フェーズでは、データにフル・アクセスできます。アップロード・フェースでは、読み込み専用アクセスが可能です。

  「Ultra Light での同時実行性」『Ultra Light - データベース管理とリファレンス』を参照してください。
- **Palm OS の強化**　Palm OS 上での Ultra Light コードの構造が再編成され、Palm データベース・セグメントを適切に使用できるようになりました。

- **エラー情報の拡張**　Ultra Light コンポーネントを使用して構築されたアプリケーションに対して、より多くのエラー情報が提供されます。
- **Windows NT/2000/XP 上で使用可能なユニコード・ライブラリ**　Embedded SQL と静的型 C++ API アプリケーション用に、ユニコード・バージョンの Ultra Light ランタイム・ライブラリが用意されています。このバージョンを使用するのは Ultra Light コンポーネントです。このライブラリを使用すると、Ultra Light データベース・ファイルは、Windows CE とデスクトップ・オペレーティング・システムとの互換性を持ちます。
- **配備プラットフォームとして Windows XP をサポート**　Windows XP への Ultra Light アプリケーションの配備がサポートされるようになりました。

バージョン 9.0 での動作の変更

この項では、SQL Anywhere Studio バージョン 9.0 のコンポーネントに導入された動作の変更について説明します。

Adaptive Server Anywhere の動作の変更

ここでは、これまでのバージョンとは異なる動作をリストにして説明します。

- **データベース内の Java オブジェクトのサポート終了** Java オブジェクトとしてのデータの保存はサポートされなくなりました。Java ストアド・プロシージャは引き続きサポートされます。

  「データベースにおける Java」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **Windows 環境用の新しいギリシャ語照合の追加** 以前のバージョンに OEM/DOS 文字セットのギリシャ語照合はありましたが、Windows 用のギリシャ語照合 1253ELL が新しく追加されました。ギリシャ語の Windows 環境でデータベースを新しく作成する場合、照合を指定しないと、1253ELL が自動的に選択されます。

  「サポートされている照合と代替照合」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しい接続制限** データベース・サーバは、接続制限より 1 つ余分に DBA 接続を許可します。これは、意図的なサービス拒否や不用意なサービス拒否があった場合に、接続してそれらの接続を切断できるようにするためです。

  「-gm サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **単一のプロセッサに限定されたパーソナル・データベース・サーバ** ソフトウェアの以前のバージョンでは、パーソナル・データベース・サーバは、要求を処理するために最大 2 つの CPU を使用していました。現在、パーソナル・サーバは単一のプロセッサに限定されています。

- **FROM 句で指定されているテーブル式への参照を、ネストしている外部ジョインの ON 句の中で使用できる** 以前のリリースでは、ON 句内の外部参照が可能でした。現在のリリースでは、このような外部参照を LATERAL キーワードで指定する必要があります。このような制限を課すことで外部参照であることが明白になり、SQL/99 規格にも準拠します。

  次のクエリは、LATERAL キーワードを使用せずに外部参照 (強調表示されている部分) を指定しているため、現在のバージョンでは正しくありません。

```
SELECT *
FROM T1,
  T2 LEFT OUTER JOIN
              ( T3 LEFT OUTER JOIN T4 ON T1.col1 = T2.col2 )
  ON T1.col2 = T2.col2
```

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

- **非修飾テーブル参照に一致するものが複数ある場合、構文エラーとして報告される**　以前のリリースでは、所有者名の指定がないテーブルへの参照 (非修飾テーブル参照) がクエリに含まれていて、そのテーブルの可能性があるものが複数ある場合、最初に見つかったものが使用されていました。現在のリリースでは、非修飾テーブル参照はエラーになります。

  「テーブル名 '%1' はあいまいです。」『SQL Anywhere 10 - エラー・メッセージ』を参照してください。

- **NULL エスケープ文字を含む LIKE 演算子は NULL に評価される**　NULL エスケープ文字を含む LIKE 演算子は NULL に評価されます。以前のリリースでは、NULL エスケープ文字を含む LIKE 演算子は、エスケープ文字がないものとして評価されていました。新しい動作は ISO/ANSI 規格に一致します。

- **プロパティと統計値の削除**　ServerIdleWaits データベース・プロパティ、TaskSwitch 接続プロパティ、CurrTaskSwitch 接続プロパティが削除されました。また、各プロパティに対応するパフォーマンス・モニタ統計値の Context Switches、Server Idle Waits/sec、Request Queue Waits/sec も削除されました。

- **INSERT/UPDATE/DELETE 実行時のカラム統計値の更新**　INSERT 文、UPDATE 文、または DELETE 文を実行して大量のデータが変更された場合、統計値が更新されるようになりました。

- **リカバリ時には統計値が更新されない**　リカバリ時、または単純な DELETE 文や UPDATE 文の実行時には、統計値は更新されなくなりました。単純な文とは、最適化されず、サーバが直接実行する文です。

- **正しいデータ型で表示されるヒストグラム範囲**　sa_get_histogram() システム・プロシージャとヒストグラム [dbhist] ユーティリティは、これまではハッシュ値で出力範囲を表示していました。現在のバージョンでは、出力されるヒストグラムの範囲は対応するカラム内のデータに一致し、正しいデータ型で表示されます。

  「ヒストグラム・ユーティリティ (dbhist)」『SQL Anywhere サーバ - データベース管理』と「sa_get_histogram システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **リモート・データベースあたり 1 人の統合ユーザのみ可能**　同じリモート・データベースに複数の統合ユーザを定義できなくなりました。

  「GRANT CONSOLIDATE 文 [SQL Remote]」『SQL Anywhere サーバ - SQL リファレンス』または「REVOKE CONSOLIDATE 文 [SQL Remote]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **明示的に指定されていない場合、CommLinks 接続パラメータは共有メモリを使用する**　CommLinks 接続パラメータが指定されていない接続は、常に共有メモリを介して接続を試みるようになりました。

  「CommLinks 接続パラメータ [LINKS]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **CommLinks 接続パラメータは、常に共有メモリ・プロトコルを最初に試行する**　CommLinks=all を指定すると、常に共有メモリ・プロトコルを使用して接続が試行され、その後に別のプロトコルを使用して接続が試行されます。

「CommLinks 接続パラメータ [LINKS]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **接続エラーによるプロセスの中止** 以前のバージョンでは、接続が確立されるまで、CommLinks 接続パラメータにリストされている接続プロトコルが 1 つずつ試行されました。現在のバージョンでは、プロセスの最中に接続エラーが発生すると、リストされているプロトコルがすべて試行されたかどうかにかかわらず、接続プロセスが即時に中止されます。

  「CommLinks 接続パラメータ [LINKS]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **prevent_article_pkey_update のデフォルト値の変更** プライマリ・キー値の更新を回避するために、prevent_article_pkey_update データベース・オプションのデフォルト値が On に変更されました。新しいデフォルト設定によって、パブリケーションに含まれるプライマリ・キーにおける、プライマリ・キーの更新は認められません。値を OFF に設定することで、この機能を無効にできます。

  「prevent_article_pkey_update オプション [データベース] [Mobile Link クライアント]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **非確定として扱われる関数** RAND 関数、NEWID 関数、GET_IDENTITY 関数は、非確定として扱われます。この結果、これらの関数はクエリの実行時にキャッシュされません。

  詳細については、「関数のキャッシュ」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **パフォーマンス・メッセージにデータベース名を表示** エンジン・パフォーマンス・アドバイス・メッセージにデータベース名が表示されるようになりました。これは、複数のデータベースを実行しているときに特に便利です。また、*Note* という文字列で始まるメッセージは、これがアドバイス・メッセージであることを表します。

- **Adaptive Server Anywhere バージョン 9.0.0 以前を使用している NetWare クライアントはアップグレードが必要** Adaptive Server Anywhere における NetWare サポートの強化の結果、Adaptive Server Anywhere バージョン 9.0.0 以前を使用している NetWare クライアントは、特定の EBF をインストールしていないかぎり、共有メモリを使用して 9.0.0 サーバに接続できません。7.0.4.3400、8.0.0.2358、8.0.1.3088、8.0.2.4095 より前のビルド番号のクライアントは、どれも 9.x サーバを検出できません。

- **ALTER DATABASE CALIBRATE の構文の変更** ALTER DATABASE CALIBRATE TEMPORARY DBSPACE の構文が、他の類似の文の構文に合わせて、ALTER DATABASE CALIBRATE DBSPACE TEMPORARY に変更されました。

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

- **積極的に行われる動的キャッシュ・サイズ決定** 新しいデータベースを起動した後、またはファイルのサイズがきわめて大きくなった場合に、動的キャッシュ・サイズ決定が積極的に行われてキャッシュのサイズを変更します。この変更が加えられる前は、最高でも 1 分間に 1 回だけ統計値がサンプリングされ、キャッシュのサイズが変更されていました。現在のバージョンでは、データベースを起動した後、またはファイルがきわめて大きくなった場合に、5 秒から 30 秒ごとに統計値がサンプリングされ、キャッシュのサイズが変更されます。

「パフォーマンス向上へのキャッシュの使用」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **インタフェースとメッセージの言語の決定** ASLANG と ASCHARSET という 2 つの環境変数が、インタフェース (Sybase Central や Interactive SQL) とメッセージで使用される言語を制御します。ASLANG は言語を指定し、ASCHARSET は文字セットを指定します。

  「SALANG 環境変数」『SQL Anywhere サーバ - データベース管理』または「SACHARSET 環境変数」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **返されるローが rowcount 設定で制限される** rowcount 設定によって、カーソルが返す先頭からのローが制限されます。現在のバージョンでは、絶対フェッチを使用して結果の先頭に移動できなくなりました。

  このような動作が必要な場合は、TOP N / START AT という新しい機能を使用することで、その動作を実現できます。

  「上位 N のソート・アルゴリズム」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

今後サポートを終了する予定の機能およびサポートを終了した機能

Adaptive Server Anywhere -d サーバ・オプションのサポート終了 Adaptive Server Anywhere の NetWare サポートが強化された結果、-d サーバ・オプションはサポートを終了しました。

NetWare 4.x のサポート終了 Adaptive Server Anywhere の NetWare サポートが強化された結果、Adaptive Server Anywhere は、NetWare バージョン 5.1 SP6 以上、またはバージョン 6.0 SP3 以上でのみ動作します。適切なサービス・パックをインストールしてください。適切なサービス・パックがインストールされていないと、Adaptive Server Anywhere サーバからエラー・メッセージが表示されます。

SQLLOCALE 環境変数のサポート終了 SQLLOCALE 環境変数に代わって、ASLANG と ASCHARSET という 2 つの新しい環境変数が使用されるようになりました。

「バージョン 9.0 での動作の変更」 224 ページを参照してください。

Mobile Link の動作の変更

ここでは、これまでのバージョンとは異なる動作をリストにして説明します。

- **dbmlsync の -i オプションと SiteScriptName 拡張オプションのサポート終了** dbmlsync -i と dbmlsync -e sc は、サポートされなくなりました。代わりに、sp_hook_dbmlsync_schema_upgrade という新しいフックが使用されます。

  「sp_hook_dbmlsync_schema_upgrade」『Mobile Link - クライアント管理』を参照してください。

- **ダウンロード確認がデフォルトでは OFF** Adaptive Server Anywhere リモートの場合、SendDownloadAck 拡張オプションのデフォルト値は OFF です。Ultra Light の場合、ul_synch_info 構造体の send_download_ack フィールドのデフォルト値は ul_false です。

バージョン 9 にアップグレードする場合、ダウンロード・トランザクションのコミット前に、リモートがダウンロードを適用したことをアプリケーションが確認する必要がある場合は、このオプションを明示的に On にしてください。次の項を参照してください。

- 「SendDownloadACK (sa) 拡張オプション」『Mobile Link - クライアント管理』
- 「Send Download Acknowledgment 同期パラメータ」『Mobile Link - クライアント管理』

- **Windows CE デバイス上でデフォルトでは動作しない dbmlsync フックがある** dbmlsync の LockTables 拡張オプションが変更され、テーブルを共有モードでロックするか、または排他モードでロックするかを指定できるようになりました。LockTables のデフォルト設定は ON であり、Windows CE 以外のすべてのプラットフォームで、テーブルを共有モードでロックし続けます。しかし、Windows CE デバイスの場合、ON は、テーブルを排他モードでロックすることを意味します。この変更によって、Windows CE アプリケーションのパフォーマンスが大幅に向上します。

  dbmlsync のイベント・フックである sp_hook_dbmlsync_download_com_error、sp_hook_dbmlsync_download_fatal_sql_error、sp_hook_dbmlsync_download_log_ri_violation はすべて、個別の接続で実行されます。これらのフックが排他モードでロックされている同期テーブルへのアクセスを試みても、正しく実行できません。Windows CE 上での配備の中でこのいずれかのフックを使用する場合は、LockTables を SHARE に設定する必要があります。次の項を参照してください。

  - 「LockTables (lt) 拡張オプション」『Mobile Link - クライアント管理』
  - 「sp_hook_dbmlsync_download_com_error (旧式)」『Mobile Link - クライアント管理』
  - 「sp_hook_dbmlsync_download_fatal_sql_error (旧式)」『Mobile Link - クライアント管理』
  - 「sp_hook_dbmlsync_download_log_ri_violation」『Mobile Link - クライアント管理』

- **Mobile Link サーバのエラー・コード** Mobile Link サーバから、エラーに関する詳しい情報が出力されるようになりました。Mobile Link サーバ・エラー・コードは -10001 から始まり、すべてが -10000 未満です。dbmlsync の場合、エラーは GUI と出力ファイルに出力されます。Ulrta Light の場合、エラーは ul_synch_info 構造体の中の文字列として有効です。

  「Mobile Link サーバのエラー・メッセージ」『SQL Anywhere 10 - エラー・メッセージ』を参照してください。

- **今後サポートを終了する予定のアップロード・カーソル** upload_cursor、new_row_cursor、old_row_cursor の各スクリプトは今後サポートを終了する予定です。アップロード・ストリームには文ベースのスクリプトを使用してください。

  「ローをアップロードするスクリプトの作成」『Mobile Link - サーバ管理』を参照してください。

- **サポート終了予定の -zac と -zec** Mobile Link サーバの、カーソルベースのスクリプトを生成する -zac と -zec オプションは廃止される予定です。

- **-zd の削除** Mobile Link サーバの、last_download タイムスタンプを最後に渡すことを指定する -zd オプションが削除されました。この結果、このパラメータは常に最初に渡されるようになりました。

- **サポート終了予定の mlxtract** mlxtract ユーティリティは今後サポートを終了する予定です。

「リモート・データベースの作成」『Mobile Link - クライアント管理』を参照してください。

- **end_synchronization スクリプトが常に呼び出される** 9.0 より以前のバージョンでは、同期が失敗したときに end_synchronization スクリプトが呼び出されないことがありました。現在のバージョンでは、begin_synchronization スクリプトが呼び出されていれば、スクリプトは常に呼び出されます。つまり、同期が成功したかどうかに関わらず、end_synchronization スクリプトに設定したクリーンアップ・アクティビティは実行されます。

  また、end_synchronization スクリプトには sync_ok という新しいパラメータが追加されています。このパラメータは、同期の成功 (1) または失敗 (0) を表します。

  「end_synchronization 接続イベント」『Mobile Link - サーバ管理』と「end_synchronization テーブル・イベント」『Mobile Link - サーバ管理』を参照してください。

- **ストリーム dll と共有オブジェクトの名前変更** ストリーム dll と共有オブジェクトの名前が変更され、Adaptive Server Anywhere との統一が図られました。次の表は、変更の詳細を示します。

以前の名前	新しい名前
dbhttp9	dbmlhttp9
dbhttps9	dbmlhttps9
dbjrsa9	dbmljrsa9
dbjtls9	dbmljtls9
dbrsa9	dbmlrsa9
dbsock9	dbmlsock9
dbtls9	dbmltls9

  「Mobile Link アプリケーションの配備」『Mobile Link - サーバ管理』を参照してください。

- **ScoutSync のサポート終了** ScoutSync はサポートされなくなりました。

- **同期のたびにスキーマ情報は再ロードされない** 9.0 より以前のバージョンでは、同期のたびにデータベースからスキーマ情報が再ロードされていました。現在のバージョンでは、dbmlsync の起動時にのみ、スキーマ情報は再ロードされます。dbmlsync -sc オプションを使用すると、以前の動作に戻すことができます。-sc オプションを使用していない場合は、スキーマ変更がリモート・データベースに加えられる前に、dbmlsync をシャット・ダウンする必要があります。シャット・ダウンせずにスキーマ変更を加えると、同期エラーが発生する可能性があります。または、dbmlsync が予期せぬ動作をすることがあります。

  「-sc オプション」『Mobile Link - クライアント管理』を参照してください。

- **主要なスクリプトがない場合は同期が中止される** 9.0 より以前のバージョンでは、特定のスクリプトがないためにデータの損失を引き起こす可能性がある場合でも、同期は続けられました。現在のバージョンでは、このような場合に Mobile Link は中止するようになりました。dbmlsrv9 -fr オプションを使用すると、失敗の代わりにエラーを出力できます。

「-fr オプション」『Mobile Link - サーバ管理』を参照してください。

- **keep_alive 同期パラメータは削除される** TCP/IP と HTTP プロトコルの keep_alive 同期パラメータは有効でなくなり、常に ON に設定されるようになりました。これは、以前のデフォルト設定でした。TCP/IP 接続の活性を制御するには、liveness_timeout パラメータを使用します。

  「CommunicationAddress (adr) 拡張オプション」『Mobile Link - クライアント管理』または「-x オプション」『Mobile Link - サーバ管理』で liveness_timeout パラメータに関する説明を参照してください。

Ultra Light の動作の変更

ここでは、これまでのバージョンとは異なる動作をリストにして説明します。

- **サポートされるプラットフォームの変更** Ultra Light 配備プラットフォームのサポートは、次のように変更されました。
  - **ScoutSync のサポート終了** ScoutSync 同期ソフトウェアはサポートされなくなりました。
  - **VxWorks のサポート終了** VxWorks オペレーティング・システムはサポートされなくなりました。
  - **pure Java Ultra Light では JDK 1.1.8 が必要** pure Java 静的型開発モデルでは、JDK 1.1.4 以上ではなく、JDK 1.1.8 以上が必要です。
  - **Palm OS の変更** Palm OS 用の Ultra Light アーキテクチャの変更によって、最新のデバイスにおけるパフォーマンスが向上します。その結果、以前のリリースよりも多くの動的メモリが必要です。きわめて小さなデータベースを除いて、Palm OS バージョン 3.5 以上と 4 MB 以上のメモリを使用することをおすすめします。
  - **MobileBuilder と PRC ツールのサポート中止** PenRight! MobileBuilder プラットフォーム上での Ultra Light の開発はサポートされなくなりました。GNU PRC Tool チェーンを使用しての開発もサポートされなくなりました。
- **開発プラットフォームの変更** Ultra Light コンポーネントのアプリケーション開発は、Windows NT/2000/XP でのみサポートされます。Windows 98 SE 上での静的インタフェースを使用しての開発もサポートされます。それ以外の Windows 95/98/Me ファミリ・メンバは、開発目的ではサポートされません。

  サポートされる Metrowerks CodeWarrior のバージョンは 8 と 9 です。
- **マニュアル内の用語の変更** Ultra Light コンポーネントの説明では、異なるインタフェースを区別するために、新しい名前が必要です。以前の Ultra Light インタフェース (Embedded SQL、C++ API、Java API) は、使用するクエリをコンパイル時に指定する必要があるために、「**静的インタフェース**」という名前が付けられています。コンポーネントは、「**動的 SQL**」へのアクセスを提供します。
- **Windows NT/2000/XP 上の Ultra Light ランタイム・ライブラリ** ActiveX と MobileVB コンポーネントは、Windows 上のユニコード・ランタイム・ライブラリを使用します。このランタイム・ライブラリは、Windows 版のバージョン 8.0.2 Ultra Light データベース (*.udb*) ファイル

と互換性がありますが、他の Windows オペレーティング・システム上で構築されたバージョン 8.0.2 Ultra Light データベース・ファイルとの互換性はありません。

- **file_name パラメータ**　ソフトウェアの以前のバージョンでは、デスクトップ上の Ultra Light データベース・ファイル名を指定するための file_name パラメータは、platform-specific パラメータが指定されていない場合は、デバイス上のファイル名を指定するためにも使用されていました。現在のバージョンでは、file_name パラメータは、デスクトップ・オペレーティング・システムを除いては無視されるようになりました。
- **静的型 Java API の変更**　静的型 Java API が変更されました。JdbcDatabase オブジェクトにあった次のメソッドが、JdbcConnection オブジェクトに移動されました。
  - countUploadRows
  - getLastDownloadTimeDate
  - getLastDownloadTimeLong

  明示的な JdbcManager オブジェクトを持たないアプリケーションが使用できるように、JdbcConnection に grant メソッドと revoke メソッドが追加されました。
- **エラー・コードの変更**　一部の Ultra Light エラー・コードが、より具体的で実用的な値に変更されました。アプリケーションの中で個々のエラー・コードをテストしている場合は、アップグレード後に新しいコードを確認してください。

  たとえば、データベースに接続するときに SQLE_DATABASE_NOT_FOUND (または同等の Ultra Light インタフェースの 1 つ) をチェックしている場合は、これを SQLE_ULTRALITE_DATABASE_NOT_FOUND に変更する必要があります。

  エラー・コードのリストについては、使用しているインタフェース内の SQL エラー・オブジェクトを参照してください。
- **Embedded SQL の UL_STORE_PARMS の変更**　UL_STORE_PARMS マクロは、EXEC SQL CONNECT 文の中で評価されるようになりました。データベースは dbinit 呼び出しの際に起動されるのではなく、接続時に起動されます。したがって、UL_STORE_PARMS は、複数の接続を使用している場合は、異なる回数だけ評価されます。また、EXEC SQL CONNECT 文の前に UL_STORE_PARMS を定義する必要もあります。

第 6 章

バージョン 8.0.2 の新機能

目次

バージョン 8.0.2 の新機能

この項では、SQL Anywhere Studio バージョン 8.0.2 のコンポーネントに導入された新機能について説明します。

Adaptive Server Anywhere の新機能

この項では、Adaptive Server Anywhere バージョン 8.0.2 の新機能について説明します。新機能について主要なものからそうでないものまですべてを示し、各機能の詳細についての参照先も記述しています。

新機能のハイライト

- **クラスタード・インデックスのサポート** テーブルにクラスタード・インデックスを作成すると、そのテーブル内のローは、インデックス内での表示順序とほぼ同じ順序で格納されます。LOAD TABLE 文を使用して、クラスタされた順序でテーブルを情報とともにロードできます。情報をテーブルに挿入すると、テーブルのクラスタの特性が劣化します。REORGANIZE TABLE 文を使用して、クラスタ順序を回復できます。クラスタード・インデックスを使用すると、パフォーマンスが向上します。

  このリリースより前に作成されたデータベースでクラスタード・インデックスを使用するには、データベースをアンロードして再ロードすることによってそのデータベースのファイル・フォーマットをアップグレードする必要があります。

  詳細については、「クラスタード・インデックスの使用」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **ユニークな識別子のサポート** Adaptive Server Anywhere は、ユニークな識別子 (UUID と GUID) をサポートします。UUID (ユニバーサル・ユニーク識別子) と GUID (グローバル・ユニーク識別子) は、同期環境内の異なるデータベースにわたってローをユニークに識別するメカニズムです。

  詳細については、「NEWID デフォルト」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **ON EXISTING 句での既存のローの更新** テーブルにプライマリ・キーがあるかぎり、INSERT 文の ON EXISTING 句を使用して既存のローを新しい値で更新できます。

  詳細については、「INSERT によるデータの変更」『SQL Anywhere サーバ - SQL の使用法』または「INSERT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **Windows CE での BACKUP 文のサポート** Adaptive Server Anywhere では、Windows CE プラットフォームで動作するデータベースのイメージ・バックアップを作成したり、データベースのトランザクション・ログの名前を変更またはトランケートしたりできます。

  詳細については、「バックアップの種類」『SQL Anywhere サーバ - データベース管理』または「BACKUP 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **グラフィカルなプランの強化** グラフィカルなプランが拡張されて、より多くの情報が含まれるようになり、外観が新しくなりました。

  詳細については、「グラフィカルなプラン」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **ワーク・テーブルの明示的な使用** プランでのワーク・テーブルができるかぎり後まで使用されないようになりました。ワーク・テーブルが使用されているときは、グラフィカルなプランに明示的に表示されます。

  詳細については、「グラフィカルなプラン」 『SQL Anywhere サーバ - SQL の使用法』または「クエリ処理におけるワーク・テーブルの使用 (all-rows 最適化ゴールの使用)」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **新しいジョインの追加** このリリースに追加された新しいジョインには、ネスト・ループ・セミジョイン、ネスト・ループ非セミジョイン、ハッシュ・セミジョイン、ハッシュ非セミジョインがあります。

  詳細については、「ジョイン・アルゴリズム」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

関数の強化

- **特定のカーソル・タイプの SQL クエリのプランを取得** PLAN、EXPLANATION、GRAPHICAL_PLAN の各関数を使用し、カーソル・タイプに基づいて SQL クエリのプランを取得できます。

  詳細については、「GRAPHICAL_PLAN 関数 [その他]」 『SQL Anywhere サーバ - SQL リファレンス』、「EXPLANATION 関数 [その他]」 『SQL Anywhere サーバ - SQL リファレンス』、または 「PLAN 関数 [その他]」 『SQL Anywhere サーバ - SQL リファレンス』 を参照してください。

  Interactive SQL でのこれらのプラン・オプションの設定については、「[オプション] ダイアログ : [プラン] タブ」 『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

- **文字セット変換関数** 新しい関数 CSCONVERT を使用して、文字列の文字セットを変換できます。

  詳細については、「CSCONVERT 関数 [文字列]」 『SQL Anywhere サーバ - SQL リファレンス』 を参照してください。

- **変数テスト関数** 新しい関数 VAREXISTS を使用して、特定の名前のユーザ定義変数が作成または宣言されているかどうかをテストできます。このテストを実行してから、必要に応じて変数を作成し、その変数を安全に使用できます。

  詳細については、「VAREXISTS 関数 [その他]」 『SQL Anywhere サーバ - SQL リファレンス』 を参照してください。

文の強化

- **プロシージャ・テキストを隠して論理の機密を保持**　SET HIDDEN オプションを使用して、ストアド・プロシージャ、関数、トリガ、ビューに含まれる論理を隠すことができます。これによって、ストアド・プロシージャ、関数、トリガ、ビュー内の論理を明らかにせずに、アプリケーションやデータベースを配布できます。

  詳細については、「プロシージャ、関数、トリガ、ビューの内容を隠す」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **LOAD TABLE で 1 バイトより長いデリミタが使用可能**　LOAD TABLE 文は、255 バイトまでのデリミタをサポートするようになりました。

  詳細については、「LOAD TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **Adaptive Server Enterprise と Microsoft SQL Server の互換性を提供する新しい文**　DEALLOCATE 文を使用して、カーソルに関連付けられているリソースを解放できます。この文は、Adaptive Server Enterprise と Microsoft SQL Server の間の互換性のために用意されています。

  詳細については、「DEALLOCATE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **dblog ユーティリティと同様の動作をする ALTER DATABASE 文**　ALTER DATABASE 文を使用して、データベース・ファイルに関連付けられているトランザクション・ログとミラー・ログの名前を変更できます。以前は、これらのログの名前の変更はトランザクション・ログ (dblog) ユーティリティでのみ行うことができました。

  詳細については、「ALTER DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **LOAD TABLE がグローバル・テンポラリ・テーブルとローカル・テンポラリ・テーブルの両方で使用可能**　Adaptive Server Anywhere は、宣言されたローカル・テンポラリ・テーブルで LOAD TABLE 文をサポートするようになりました。以前は、グローバル・テンポラリ・テーブルのみがサポートされていました。

  詳細については、「LOAD TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **SET 文を使用した変数値の割り当てが可能**　Transact-SQL プロシージャで SET 文を使用して、変数に値を割り当てることができるようになりました。

- **INSERT 文での WITH AUTO NAME のサポート**　INSERT 文で WITH AUTO NAME を指定すると、SELECT リスト内の項目の名前によって、値と挿入先カラムの関連付けが決定されます。

  詳細については、「INSERT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **EXIT 文の強化**　Interactive SQL の EXIT 文は、Interactive SQL に終了コードを設定できるようになりました。

詳細については、「EXIT 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **FROM 句のクエリに対する最適化ゴールの指定** FASTFIRSTROW テーブル・ヒントを使用すると、optimization_goal オプションを first-row に設定しなくてもクエリの最適化ゴールを設定できます。

  詳細については、「FROM 句」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

セキュリティの強化

- **新しいユーティリティを使用してファイルの内容を隠す** 設定ファイル (別名コマンド・ファイル) には、パスワードが含まれることがあります。強化されたセキュリティ機能として、Adaptive Server Anywhere にはファイル非表示ユーティリティという新しいユーティリティがあります。このユーティリティを使用すると、単純な暗号で設定ファイルの内容を隠すことができます。

  詳細については、「ファイル非表示ユーティリティ (dbfhide)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Certicom 暗号化の変更** ECC_TLS と RSA_TLS という 2 種類の Certicom 暗号化をサポートするようにセキュリティが強化されました。以前のバージョンの Adaptive Server Anywhere で Certicom 暗号化と呼ばれていた暗号化は、ECC_TLS 暗号化という名前に変更されました。Certicom パラメータは現在も受け入れられますが、ECC_TLS 暗号化と同等です。Adaptive Server Anywhere は、RSA_TLS 暗号化もサポートするようになりました。

  詳細については、「-ec サーバ・オプション」『SQL Anywhere サーバ - データベース管理』または「Encryption 接続パラメータ [ENC]」『SQL Anywhere サーバ - データベース管理』を参照してください。

パフォーマンスの強化

- **新しい接続パラメータによるネットワークの応答性の向上** LazyClose と PrefetchOnOpen ネットワーク接続パラメータを使用すると、遅延時間の長いネットワークまたは多くの要求を処理するアプリケーションのあるネットワークで、パフォーマンスが向上します。

  これらのパラメータについては、「LazyClose 接続パラメータ [LCLOSE]」『SQL Anywhere サーバ - データベース管理』と「PrefetchOnOpen 接続パラメータ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Windows NT/2000/XP で分散読み込みが可能** 以前は、大きなテーブルの逐次スキャンでは、ページが 64 KB バッファにコピーされてからキャッシュにコピーされていました。現在は、Windows NT サービス・パック 2 以上または Windows 2000/XP の環境で実行し、ページ・サイズが 4 KB 以上であれば、分散読み込みによってページがキャッシュに直接コピーされるため、時間が節約され、パフォーマンスが向上します。

  詳細については、「適切なページ・サイズの使用」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **要求のロギングでの時間単位の向上** プロシージャ・プロファイリングまたは要求のロギングを使用して取得される時間は、1 ミリ秒の単位になりました。この変更は、主に Windows オペレーティング・システムで稼働しているサーバに影響します。
- **複数バージョンのパフォーマンス・モニタの実行** 複数バージョンの Adaptive Server Anywhere を同時に実行している場合は、複数バージョンの Windows パフォーマンス・モニタも同時に実行できます。

  Windows パフォーマンス・モニタの詳細については、「Windows パフォーマンス・モニタを使用した統計値のモニタリング」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

その他の機能強化

- **レジストリ設定によるサーバの temp フォルダの変更** Windows CE プラットフォームでは、レジストリを使用して、サーバが使用するテンポラリ・ディレクトリを指定できます。

  詳細については、「Windows CE でのレジストリ設定」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **新しい iAnywhere JDBC ドライバ** 信頼性とパフォーマンスに優れたこの JDBC ドライバは、ODBC データ・ソースと Command Sequence クライアント／サーバ・プロトコルの利点を利用します。これは、jConnect JDBC ドライバに代わるものです。

  iAnywhere JDBC ドライバの詳細については、「iAnywhere JDBC ドライバの使用」『SQL Anywhere サーバ - プログラミング』を参照してください。

  JDBC ドライバの選択については、「JDBC ドライバの選択」『SQL Anywhere サーバ - プログラミング』を参照してください。
- **トリガは、トリガが起動する原因となったアクションを識別できる** トリガが UPDATE、INSERT、DELETE のいずれの操作によって起動したかに応じて、異なるアクションを実行できるようになりました。この機能によって、単一トリガ内の異なるイベント間で論理を共有できるとともに、アクションに固有の方法でアクションを実行できます。

  詳細については、「トリガ・オペレーション条件」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **TDS 固有でなくなった return_date_time_as_string** すべての接続で、return_date_time_as_string オプションを使用できるようになりました。

  このオプションの詳細については、「return_date_time_as_string オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **DB 領域への領域追加時に単位の指定が可能** 特定のサイズ (ページ、キロバイト、メガバイト、ギガバイト、またはテラバイトの単位) でデータベース・ファイルを拡張できます。

  詳細については、「ALTER DBSPACE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **sa_make_object システム・プロシージャ** このシステム・プロシージャを SQL スクリプトで使用して、オブジェクトの骨格となるインスタンスが存在することを確認できます。その後で、ALTER 文を実行して実際の定義を提供できます。

  詳細については、「sa_make_object システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **Microsoft SQL Server と互換性のある新しいグローバル変数** 新しいグローバル変数が導入され、Microsoft SQL Server との互換性が提供されています。@@fetch_status グローバル変数は @@sqlstatus グローバル変数と同じですが、最後のフェッチのステータスを別の値で返す点が異なります。

  詳細については、「グローバル変数」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **NetWare でサポートされる文字セット変換** NetWare は、文字セット変換をサポートするようになりました。

- **インストールされている Java クラスのバージョンを情報ユーティリティがレポート** dbinfo ユーティリティと a_db_info 構造体は、データベースにインストールされている Java クラスのバージョンをレポートするようになりました。

  詳細については、「情報ユーティリティ (dbinfo)」『SQL Anywhere サーバ - データベース管理』と「a_db_info 構造体」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **フェッチ操作の警告の抑制** 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。[Adaptive Server Anywhere 9 の ODBC 設定] ダイアログでは、データベース・サーバから返される警告メッセージを抑制し、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、警告メッセージが適切に処理されるようにできます。

  詳細については、「[ODBC 設定] ダイアログ : [ODBC] タブ」『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

- **プライマリ・キー・カラムの更新の制御** 新しい prevent_article_pkey_update オプションを On に設定すると、パブリケーションに含まれるテーブルのプライマリ・キー・カラムの更新が禁止されます。このオプションを使用することで、特にレプリケーションと同期の環境におけるデータの整合性が確保されます。

  詳細については、「prevent_article_pkey_update オプション [データベース] [Mobile Link クライアント]」『SQL Anywhere サーバ - データベース管理』を参照してください。

Mobile Link の新機能

次に、バージョン 8.0.2 で導入したソフトウェアに加えられた変更と追加を示します。

- **.NET のサポート** Mobile Link では、同期スクリプトで Visual Studio .NET プログラミング言語の記述がサポートされるようになりました。

詳細については、「.NET での同期スクリプトの作成」『Mobile Link - サーバ管理』、「-sl dnet オプション」『Mobile Link - サーバ管理』、「ml_add_dnet_connection_script」『Mobile Link - サーバ管理』、および「ml_add_dnet_table_script」『Mobile Link - サーバ管理』を参照してください。

- **起動クラス** 最初の同期の前、Mobile Link サーバの起動時に Java 仮想マシンまたは CLR を実行する Java コードと .NET コードを記述できるようになりました。

  詳細については、「ユーザ定義起動クラス」『Mobile Link - サーバ管理』を参照してください。

- **UUID を使用したユニークなプライマリ・キーの管理** リモート・データベースでユニークなプライマリ・キーを管理する新しい方法が、ユニバーサル・ユニーク ID (UUID、別名 GUID) とともに導入されました。

  詳細については、「UUID の使用」『Mobile Link - サーバ管理』を参照してください。

- **参照整合性違反の新しい処理方法** ダウンロード時の参照整合性違反の管理に役立つ 2 つの新しいクライアント・イベント・フック sp_hook_dbmlsync_download_ri_conflict と sp_hook_dbmlsync_download_log_ri_conflict が導入されました。

  詳細については、「sp_hook_dbmlsync_download_ri_violation」『Mobile Link - クライアント管理』と「sp_hook_dbmlsync_download_log_ri_violation」『Mobile Link - クライアント管理』を参照してください。

- **リモート・テーブルのすべてのローを削除する簡単な方法** すべてのプライマリ・キー・カラムが NULL である 1 つのローを download_delete_cursor に含めると、リモート・テーブル内のすべてのデータを削除できます。

  詳細については、「download_delete_cursor スクリプトの作成」『Mobile Link - サーバ管理』を参照してください。

パフォーマンスとモニタリングの強化

- **Mobile Link モニタ** グラフィカル・ツールである Mobile Link モニタが導入され、各同期処理にかかった時間を Mobile Link ユーザまたはワーカ・スレッドでソートして表示できるようになりました。

  詳細については、「Mobile Link モニタ」『Mobile Link - サーバ管理』を参照してください。

- **dbmlsync にアップロードされるロー数の推定が可能** 新しい dbmlsync コマンド・ライン・オプション -urc が作成されました。アップロードされるロー数を推定することで、同期のパフォーマンスを改善できます。

  詳細については、「-urc オプション」『Mobile Link - クライアント管理』を参照してください。

- **HTTP/HTTPS 接続の継続を指定可能** **persistent** オプションを使用して、同期のすべての HTTP 要求に同じ接続を使用するように Mobile Link に指定できます。この設定によって、パフォーマンスを向上できます。この設定は、Mobile Link に直接接続しているときにだけ使用してく

ださい。プロキシやリダイレクタなどの中間エージェントを介して接続するときには使用しないでください。

詳細については、「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **警告メッセージの新しい制御方法** 3 つの新しい dbmlsrv9 コマンド・ライン・オプション -zw、-zwd、-zwe が作成されました。-zw を使用すると、レポートする警告メッセージのレベルを制御できます。-zwd を使用すると、特定の警告コードを無効にできます。-zwe を使用すると、-zw で無効にされている特定の警告コードを有効にできます。

  詳細については、「-zw オプション」『Mobile Link - サーバ管理』、「-zwd オプション」『Mobile Link - サーバ管理』、および「-zwe オプション」『Mobile Link - サーバ管理』を参照してください。

- **新しい冗長ロギング・オプション** dbmlsync -v コマンド・ライン・オプションが変更され、拡張されました。現在は、-v を単独で使用すると冗長性が最小になります。冗長性を最大にするには、-v+ を使用します。ログに記録された情報を詳細に調整するために指定できる新しいレベルもいくつかあります。これらのオプションは、拡張オプションとしても使用できます。

  詳細については、「-v オプション」『Mobile Link - クライアント管理』を参照してください。

接続の強化

- **ping のサポート** リモート・データベースは、Mobile Link サーバに対して ping を実行できるようになりました。

  詳細については、「-pi オプション」『Mobile Link - クライアント管理』と「Ping 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。

- **新しい同期ストリーム** Mobile Link では、HTTPS プロトコルがサポートされるようになりました。この新しいストリームは、RSA 暗号化を使用して HTTP over SSL/TLS を実装します。また、他の HTTPS サーバと互換性があります。

  詳細については、「-x オプション」『Mobile Link - サーバ管理』と「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **新しい buffer_size オプション** **buffer_size** オプションを使用して、固定長の HTTP メッセージの最大バッファ・サイズを指定できるようになりました。

  詳細については、「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **Mobile Link クライアントの自動ダイヤル** Pocket PC 2002 または Windows デスクトップ・コンピュータで稼働する Mobile Link クライアントは、ダイヤルアップ・ネットワーク接続を通じて接続できるようになりました。スケジュールを使用すると、リモート側で自動的に同期を実行できます。新しい同期ストリーム・パラメータは、**network_name**、**network_connect_timeout**、**network_leave_open** です。

詳細については、「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

新しい Web サーバのサポート

- **サーブレット・リダイレクタ** Mobile Link では、Apache Tomcat など、Java サーブレット API 2.2 をサポートする Web サーバがサポートされるようになりました。

  詳細については、「リダイレクタによる Web サーバを経由した同期」『Mobile Link - サーバ管理』を参照してください。

セキュリティの強化

- **RSA 暗号パッケージ・プログラムのサポート** 既存の楕円曲線暗号化に加え、RSA 暗号化を同期セキュリティに使用できるようになりました。gencert ユーティリティと readcert ユーティリティでは、RSA 証明書と楕円曲線証明書がサポートされます。

  詳細については、「Mobile Link クライアント／サーバ通信の暗号化」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **gencert による生成済み証明書要求への署名が可能** 証明書生成ユーティリティ gencert には、生成済み証明書要求に署名できる新しいコマンド・ライン・オプションがあります。

  詳細については、「証明書生成ユーティリティ [gencert] (旧式)」『SQL Anywhere サーバ - データベース管理』を参照してください。

SQL Remote の新機能

SQL Remote バージョン 8.0.2 に、次の新機能が追加されています。

- **統合データベースに送信されるエラー・ログ** リモート・サイトでのエラーのトラブルシューティングを向上するために、統合データベースでログ情報を収集できます。

  詳細については、「リモート・サイトでのトラブルシューティング・エラー」『SQL Remote』を参照してください。

Ultra Light の新機能

Ultra Light 8.0.2 には、次の新機能が導入されています。

- **Ultra Light コンポーネント** Ultra Light データベース・テクノロジを、使いやすい方法で新しい開発プラットフォームから使用できるようになりました。Ultra Light コンポーネントは、eMbedded Visual Basic、AppForge MobileVB、Java のユーザに Ultra Light テクノロジを提供します。Java のコンポーネントは、このマニュアルで説明する Ultra Light for Java の代わりに使用できます。このコンポーネントは 100% pure Java の実装ではありませんが、パフォーマンスを向上させるためにネイティブ・クラスを使用します。

Ultra Light コンポーネントのマニュアルはオンライン・マニュアルで提供されています。まずは、Ultra Light - データベース管理とリファレンス『Ultra Light - データベース管理とリファレンス』を参照してください。

- **Ultra Light データベースのアップグレード** 新しいバージョンのアプリケーションを配備するときに、Ultra Light データベースのスキーマを新しいアプリケーションのスキーマにアップグレードするかどうかを選択できるようになりました。

  9.0.1 では、ULEnableGenericSchema は ULRegisterSchemaUpgradeObserver に置き換わっています。

- **スレッド対応の Java ランタイム** Ultra Light Java ランタイムはスレッド対応になり、マルチスレッドの Ultra Light アプリケーションを開発できるようになりました。

- **Ultra Light データベース・ファイルの削除** ULDropDatabase 関数を使用して、Ultra Light データベース・ファイルをアプリケーションから削除できます。

  詳細については、次の項を参照してください。

  - Embedded SQL：「ULDropDatabase 関数」『Ultra Light - C/C++ プログラミング』
  - C++ API：Drop メソッド

- **ユニバーサル・ユニーク識別子** Ultra Light データベースで、Adaptive Server Anywhere の UNIQUEIDENTIFIER データ型を使用できるようになりました。このデータ型は、ユニバーサル・ユニーク識別子 (UUID または GUID) の格納に使用される BINARY(16) です。NEWID 関数をデフォルト値として使用する UNIQUEIDENTIFIER カラムは、GLOBAL AUTOINCREMENT の代わりに使用でき、インストールされている Mobile Link 全体を通じてプライマリ・キーがユニークであることを保証します。

  詳細については、「NEWID デフォルト」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **同期の新しいセキュリティ・オプション** このリリースでは、2 つの新しいセキュリティ機能のある同期プロトコルが導入されました。HTTPS は、トランスポート・レイヤ・セキュリティ・プロトコルで実装した HTTP です。RSA は、HTTP または TCP/IP ネットワーク上で使用するトランスポート・レイヤ・セキュリティ暗号化の 1 つの形式です。

  これらのセキュリティ・オプションでは、Certicom テクノロジが使用されます。Certicom テクノロジを使用するには、別途ライセンスの 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。このオプションの詳細については、「SQL Anywhere 10 のコンポーネント」『SQL Anywhere 10 - 紹介』を参照してください。

  HTTPS 同期の詳細については、「Stream Type 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。

- **最終ダウンロード時間のリセット** たとえばアプリケーションをクリーンな状態に設定するために、以前にダウンロードしたデータを再同期するには、最終ダウンロード・タイムスタンプをリセットします。

詳細については、「ULResetLastDownloadTime 関数」『Ultra Light - C/C++ プログラミング』を参照してください。

- **前回の同期のトラブルシューティング**　前回の同期の成功または失敗に関する情報を取得する関数を使用できるようになりました。この機能は、HotSync を使用する Palm OS アプリケーションで、同期がアプリケーションの外部で実行される場合に特に役立ちます。

  詳細については、「ULGetSynchResult 関数」『Ultra Light - C/C++ プログラミング』を参照してください。この機能は Ultra Light Java アプリケーションにはまだ使用できません。

- **サイズの小さいファイルをたくさん生成**　-x オプションを使用すると、Ultra Light ジェネレータは、C/C++ プロジェクトについて、ファイルサイズが小さめなファイルをたくさん書き出します。このオプションは、生成されたコードが大きすぎてコンパイラが単一のファイルで処理できない場合に役立ちます。

- **強化された同期 observer**　同期 observer 関数が拡張されました。より多くのステータスとフィールドがインタフェースに追加され、応答性に優れ、情報の多い同期ダイアログを設計できます。

バージョン 8.0.2 での動作の変更

この項では、SQL Anywhere Studio バージョン 8.0.2 のコンポーネントに導入された動作の変更について説明します。

Adaptive Server Anywhere の動作の変更

ここでは、これまでのバージョンとは異なる動作をリストにして説明します。

- **Windows CE 2.11 のサポート終了**　Windows CE 2.11 プラットフォームはサポートされなくなりました。
- **SH3 チップと SH4 チップのサポート終了**　SH3 チップと SH4 チップを使用する Windows CE デバイスはサポートされなくなりました。
- **optimization_goal 設定**　optimization_goal オプションのデフォルト設定は、**first-row** ではなく **All-rows** です。これは、一部のクエリで選択されている実行プランに影響するため、パフォーマンス特性が変わります。

  詳細については、「optimization_goal オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **Windows オペレーティング・システムでの xp_cmdshell によるコマンド・ウィンドウの表示**　xp_cmdshell が新しいウィンドウを起動するかどうかを制御できるようになりました。この動作変更は、バージョン 8.0.2 以降で作成またはアップグレードされたデータベースに適用されます。それより古いデータベースの場合は、コマンド・ウィンドウを表示しない以前の動作が維持されます。この新しい動作は、Adaptive Server Enterprise や Microsoft SQL Server などの他のデータベースと互換性があります。

  xp_cmdshell の呼び出しで 2 番目のパラメータを指定すると、コマンド・ウィンドウを非表示にできます。

  詳細については、「xp_cmdshell システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **データベース・サーバで使用されている言語に関係なく、完全なスペルの英語の曜日名が認識される**　イベントを作成するときに、データベース・サーバが使用している言語 (ドイツ語、中国語など) に関係なく、完全なスペルの英語の曜日名がデータベース・サーバによって認識されます。したがって、再ロード・スクリプトのイベント定義が、異なる言語で実行されているサーバで認識されます。

  省略形の英語の曜日名 (Mon、Tue など) を使用するイベントは、英語以外の言語で実行されているサーバでは認識されません。

  詳細については、「CREATE EVENT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **オプション設定の検証**　最小値と最大値を持つ整数オプションが検証されるようになりました。オプションを無効な値に設定すると、「不正なオプション '%1' を設定しています。」『SQL Anywhere 10 - エラー・メッセージ』というエラーが発生します。

  無効なオプション設定を含むデータベースをアンロードして再ロードした場合は、最も近い有効値に設定されます。

  影響するオプションを次に示します。角カッコは上限値と下限値を示します。

オプション	範囲
isolation_level	[0, 3]
precision	[0, 127]
scale	[0, 127]
nearest_century	[0, 100]
max_hash_size	[2, 64]
MAX_WORK_TABLE_HASH_SIIZE	[2, 64]
first_day_of_week	[1, 7]
default_timestamp_increment	[1, 60000000]

- **名前変更されたジョイン**　グラフィカル・プランとドキュメントの両方で 2 つのジョインの名前が変更されました。ネスト・ループ・ジョイン NOT EXISTS (JNE) は、ネスト・ループ非セミジョイン (JNLA) と呼ばれるようになり、ネスト・ループ・ジョイン EXISTS (JE) はネスト・ループ・セミジョイン (JNLS) と呼ばれるようになりました。

  詳細については、「ジョイン・アルゴリズム」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

廃止されサポートされなくなった機能

このリストには、サポートを終了した機能の中で既存のアプリケーションに影響のあるものがまとめられています。

- **Windows で使用されなくなった -d サーバ・オプション**　-d オプションを NetWare で使用すると、DFS (Direct File System) I/O ではなく POSIX I/O が強制的に使用されます。Windows では、このオプションはコマンド・ラインで使用できますが、無視されます。

Mobile Link の動作の変更

ここでは、これまでのバージョンとは異なる動作をリストにして説明します。

- **シリアル通信プロトコルのサポート終了**　シリアル・プロトコルはサポートされなくなりました。代わりに、HTTP、HTTPS、または TCP/IP を使用できます。

- **Certicom からの証明書の発行終了**　Certicom からトランスポート・レイヤ・セキュリティ証明書を取得できなくなりました。ただし、Certicom reqtool ユーティリティを使用して証明書要求を生成し、VeriSign や Entrust Technologies などの他のさまざまなソースから証明書を購入することはできます。

  詳細については、http://www.verisign.com/ または http://www.entrust.com/certificate_services/index.htm を参照してください。

- **dbmlsrv の使用されなくなったオプション -vw**　警告メッセージを抑制するために使用していた -vw dbmlsrv コマンド・ライン・オプションは使用されなくなりました。代わりに、-zw または -zwd を使用できます。

  詳細については、「-zw オプション」『Mobile Link - サーバ管理』と「-zwd オプション」『Mobile Link - サーバ管理』を参照してください。

- **dbmlsync オプション -v の動作変更**　-v dbmlsync コマンド・ライン・オプションが変更され、拡張されました。現在は、-v を単独で使用すると冗長性が最小になります。

  詳細については、「-v オプション」『Mobile Link - クライアント管理』を参照してください。

- **同期サーバで使用されている言語に関係なく完全な長さの英語の曜日名が認識される**　Mobile Link のユーザ、パブリケーション、サブスクリプションのスケジュールを作成するとき、または dbmlsync コマンド・ラインでスケジュール情報を指定する場合、英語以外の言語で実行している同期サーバでスケジュールを認識するには、完全な長さの形式の英語の曜日名 (Monday など) を使用してください。

  省略形の英語の曜日名 (Mon など) を使用するスケジュールは、英語以外の言語で実行されている同期サーバでは認識されません。

  詳細については、「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **dbmlsync での長いデータのサポート強化**　DBMLSync は、アップロード・ストリームを構築するときに、はるかに効率的な方法で BLOB を処理するようになりました。BLOB は、断片化されてメモリに読み込まれるため、長い BLOB の処理能力が使用可能メモリによって制限されなくなりました。一度に複数のパブリケーションの同期を行う場合は、BLOB データが一度に格納され、アップロード・ストリーム間で共有されます。出力ログは、BLOB のサイズとその最初の 32 バイトを出力するようになりました。

- **HTTP オプション use_cookies の削除**　use_cookies オプションは削除されました。このオプションを設定した場合は無視されます。Mobile Link では、cookie が必要な状況が自動的に検出されます。

Ultra Light の動作の変更

ここでは、これまでのバージョンとは異なる動作をリストにして説明します。

- **Windows CE 2.11 のサポート終了**　Windows CE 2.11 プラットフォームはサポートされなくなりました。
- **SH3 チップと SH4 チップのサポート終了**　SH3 チップと SH4 チップを使用する Windows CE デバイスはサポートされなくなりました。
- **シリアル通信プロトコルのサポート終了**　シリアル・プロトコルはサポートされなくなりました。シリアル同期は、主に Palm Computing Platform 上のクライアントから使用されていました。これらのクライアントでは、代わりに HotSync 同期を使用できます。
- **VxWorks でのトランスポート・レイヤ・セキュリティなし**　同期にトランスポート・レイヤ・セキュリティを提供する Certicom ライブラリは、VxWorks オペレーティング・システムでサポートされなくなりました。
- **VxWorks 5.5 はサポートされない**　サポートされている VxWorks オペレーティング・システムのバージョンは VxWorks 5.3 と 5.4 です。

バージョン 9 における VxWorks の非サポート
バージョン 9 では VxWorks プラットフォームはいっさいサポートされません。

- **Certicom ライブラリには JDK 1.2 が必要**　このリリースでは Certicom セキュリティ・ライブラリが更新されました。Java アプリケーション用の新しいライブラリには、JDK 1.1.4 ではなく JDK 1.2 が必要です。

第 7 章

バージョン 8.0.1 の新機能

目次

バージョン 8.0.1 の新機能

この項では、SQL Anywhere Studio バージョン 8.0.1 のコンポーネントに導入された新機能について説明します。

Adaptive Server Anywhere の新機能

この項では、Adaptive Server Anywhere バージョン 8.0.1 の新機能について説明します。新機能について主要なものからそうでないものまですべてを示し、各機能の詳細についての参照先も記述しています。

- **テーブル・ページに予約する領域の指定** テーブル・ページで予約領域にする空き領域の割合を指定すると、テーブルの断片化を減らすことができます。

  詳細については、「テーブルの断片化削減」『SQL Anywhere サーバ - SQL の使用法』と「ALTER TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

このリリースより前に作成されたデータベースに対して割り付ける領域の割合を指定するには、データベースをアンロードして再ロードすることによってそのデータベースのファイル・フォーマットをアップグレードする必要があります。

- **新しいシステム・テーブル** 2 つの新しいシステム・テーブル SYSATTRIBUTE と SYSATTRIBUTENAME が追加されました。

  詳細については、「ISYSATTRIBUTE システム・テーブル」『SQL Anywhere サーバ - SQL リファレンス』と「ISYSATTRIBUTENAME システム・テーブル」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **sa_disk_free_space システム・プロシージャ** このプロシージャを使用すると、DB 領域、テンポラリ・ファイル、トランザクション・ログ、トランザクション・ログ・ミラーで使用可能な領域を決定できます。

  詳細については、「sa_disk_free_space システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **sa_flush_statistics システム・プロシージャ** データベース管理者は、このプロシージャを使用することで、データベース・サーバのキャッシュのみに存在するコスト・モデル統計情報を確実にフラッシュできます。

  詳細については、「sa_flush_statistics システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **サーバのメッセージ・ウィンドウの内容を取得する新しい方法** サーバ・メッセージ・ウィンドウから情報を返す新しいシステム・プロシージャと新しい 3 つのプロパティが用意されました。

詳細については、「sa_get_server_messages システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』と、「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』の MessageText、MessageTime、MessageWindowSize を参照してください。

- **ANSI 以外の文と等価な ANSI 文の確定** REWRITE 関数は、新しい引数 ANSI を受け付けるようになりました。これにより、REWRITE 関数は SELECT 文、UPDATE 文、または DELETE 文に対して等価な ANSI 文を返します。

  詳細については、「REWRITE 関数 [その他]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **UPDATE 文が変数への代入を許可** UPDATE 文の SET 句は、テーブルの更新だけでなく、変数への値の代入にも使用できるようになりました。この機能は、Adaptive Server Enterprise と互換性があります。

  詳細については、「UPDATE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **オートインクリメントの代替手段** GET_IDENTITY 関数は、オートインクリメント・カラムへの ID 番号割り付けの代替手段として提供されます。

  詳細については、「GET_IDENTITY 関数 [その他]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **識別子を角カッコで区切る** 識別子は、角カッコで区切ることができます。角カッコは、quoted_identifier オプションの設定に関わらず、どのような場合でも使用できます。

  詳細については、「識別子」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **FROM 句での独立性レベルの設定** WITH table-hint 引数を使用すると、特定の SELECT 文、UPDATE 文、または DELETE 文に対してテーブルまたはビューのロック方法を指定できます。

  詳細については、「FROM 句」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **[データベース移行] ウィザード** [データベース移行] ウィザードを使用すると、Sybase Central から Adaptive Server Anywhere データベースにリモート・テーブルをマイグレートできます。

  ターゲット・データベースがバージョン 8.0.0 またはそれよりも前の場合、外部キーはマイグレートできません。外部キーをマイグレートするには、ターゲット・データベースをアンロードして再ロードすることにより、データベースのファイル・フォーマットをアップグレードする必要があります。

  詳細については、「SQL Anywhere へのデータベースの移行」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **Sybase Central からのバージョン 5.x または 6.x のデータベースのアンロード** Sybase Central では、バージョン 5.x または 6.x のデータベースに接続し、[データベース・アンロード] ウィザードを使用してデータベース・ファイル・フォーマットをアップグレードできるようにな

りました。これを行うには、バージョン 8.0.0 以降のサーバ上でデータベースを実行する必要があります。

- **[データベース・アップグレード] ウィザードでのデータベースのバックアップとシャットダウン**　Sybase Central の [データベース・アップグレード] ウィザードを使用して、メインのデータベース・ファイル、トランザクション・ログ、DB 領域など、すべてのデータベース・ファイルをバックアップできるようになりました。このウィザードでは、アップグレードの完了時にデータベースをシャットダウンすることもできます。

- **sa_migrate の強化**　sa_migrate プロシージャにはオプションの引数 *migrate_fkeys* が追加されています。この引数でリモート・データベースからテーブルをマイグレートするときに、外部キー・マッピングをマイグレートするかどうかを指定できます。これまでのリリースでは、sa_migrate プロシージャを使用すると、外部キー・マッピングは常にマイグレートされていました。

  詳細については、「sa_migrate システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  このリリースより前に作成されたデータベースでこの機能を使用するには、データベースをアンロードして再ロードすることによってそのデータベースのファイル・フォーマットをアップグレードする必要があります。

- **新しい sort_collation データベース・オプション**　sort_collation データベース・オプションを使用すると、ORDER BY 式に対して SORTKEY 関数が暗黙的に使用されます。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。

  詳細については、「sort_collation オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **IP アドレス／ポートを使用してサーバに接続**　VerifyServerName=NO プロトコル・オプションを使用すると、IP アドレスかポートさえ分かっていれば、サーバ名の検証を行わずに Adaptive Server Anywhere クライアントから Adaptive Server Anywhere サーバに接続できます。VerifyServerName パラメータは、DoBroadcast=NONE を指定した場合にのみ使用します。

  詳細については、「VerifyServerName プロトコル・オプション [VERIFY]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しい LocalOnly プロトコル・オプションによるブロードキャストの管理**　LocalOnly プロトコル・オプションを使用すると、ローカル・コンピュータ上のサーバ (存在する場合) のみに接続できます。LocalOnly=YES を設定すると、サーバから他のコンピュータへのブロードキャスト応答が無視された場合を除き、通常のブロードキャスト・メカニズムが使用されます。

  詳細については、「LocalOnly プロトコル・オプション [LOCAL]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **カーソルの固定に使用するキャッシュの割合の指定**　pinned_cursor_percent_of_cache オプションを使用すると、カーソルを固定するために使用するキャッシュの割合を指定できます。この制限値を低く設定するとメモリが少ない環境のパフォーマンスが向上します。

詳細については、「pinned_cursor_percent_of_cache オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **データベース・ファイルとログ・ファイルの断片化をモニタ** DBFileFragments と LogFileFragments データベース・プロパティを使用すると、ファイルの断片化をモニタできます。通常、トランザクション・ログ・ファイルの断片化は大きな問題にはなりませんが、データベース・ファイルの断片化はパフォーマンス低下の原因となることがあるので、その場合にはディスク断片化解除ユーティリティを使用してください。

  詳細については、「データベース・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しい接続プロパティ** 2 つの接続プロパティが新しく追加されました。LivenessTimeout は接続の活性タイムアウト、IdleTimeout は接続のアイドル・タイムアウトをそれぞれ返します。

  詳細については、「接続レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しいサーバ・プロパティ** 新しい IdleTimeout サーバ・プロパティは、デフォルトのアイドル・タイムアウト値を返します。

  詳細については、「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **非決定的関数** 基本データを修正する関数、またはクエリの実行中に変更される可能性のある基本データに依存する関数を NOT DETERMINISTIC と宣言できます。このように宣言された関数は、クエリ実行中に呼び出されるたびに再評価されます。このように宣言されない関数では、パフォーマンス向上のために関数の値がキャッシュされ、再利用されます。

  詳細については、「CREATE FUNCTION 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **バックアップで、すべてのトランザクションが完了するのを待つ** デフォルトで、BACKUP 文では、開いているトランザクションが完了するのを待たずに、トランザクション・ログ名の変更やトランザクション・ログのトランケートが行われます。WAIT AFTER END 句を指定すると、バックアップに含まれるすべてのトランザクションを確実に完了できるようになりました。

  詳細については、「BACKUP 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

Mobile Link の新機能

次に、バージョン 8.0.1 で導入したソフトウェアに加えられた変更と追加を示します。

- **完全なエラー・コンテキストの報告** 同期中にエラーが発生した場合、Mobile Link サーバは、出力ファイルに完全なエラー・コンテキストを示すようになりました。

  詳細については、「-o オプション」『Mobile Link - サーバ管理』を参照してください。

- **ユーザ ID マッピング**　Mobile Link では、データベース・ユーザ ID の検索や、Mobile Link ユーザ名のユーザ ID へのマッピングが簡単になりました。

  詳細については、「modify_user 接続イベント」『Mobile Link - サーバ管理』を参照してください。

- **クライアント・オプションとしてアドレスとタイプを設定**　Mobile Link クライアントでは、コマンド・ラインで通信タイプとアドレスを指定して Mobile Link サーバに接続できるようになりました。

  詳細については、「Mobile Link SQL Anywhere クライアントの拡張オプション」『Mobile Link - クライアント管理』を参照してください。

- **Mobile Link が発行する ODBC 文のログ**　Mobile Link に対しては、Mobile Link が発行するすべての ODBC 文について ODBC 出力ファイルにログを取るように指定できます。

- **ダウンロード・タイムスタンプの修正**　2 つの新しいイベントで、最終ダウンロード・タイムスタンプまたは次の最終ダウンロード・タイムスタンプを修正できます。

  詳細については、「modify_last_download_timestamp 接続イベント」『Mobile Link - サーバ管理』と「modify_next_last_download_timestamp 接続イベント」『Mobile Link - サーバ管理』を参照してください。

- **タイムスタンプ競合への自動許容**　このオプションを使用すると、統合データベースとリモート・データベース間でタイムスタンプの競合が発生した場合、競合を検出するために、最小精度よりも高い精度のタイムスタンプ値を使用できます。

  詳細については、「-zp オプション」『Mobile Link - サーバ管理』を参照してください。

SQL Remote の新機能

SQL Remote バージョン 8.0.1 に、次の新機能が追加されています。

- **SMTP ユーザ認証**　SMTP/POP メッセージ・システムを使用しているときに、SMTP サーバでの別個のユーザ認証のためのパラメータが用意されています。

  詳細については、「SMTP メッセージ・システム」『SQL Remote』を参照してください。

Ultra Light の新機能

Ultra Light 8.0.1 には、次の新機能が導入されています。

- **CodeWarrior 8 のサポート**　このリリースでは、CodeWarrior バージョン 8 をサポートします。
- **マルチスレッド・アプリケーションのサポート**　Ultra Light アプリケーションは、マルチスレッド・アプリケーションをサポートするプラットフォーム上でマルチスレッド化できるようになりました。

- **Pocket PC 2002 のサポート**　Pocket PC 2002 が、サポートされるプラットフォームのリストに追加されています。
- **JDBC ResultSet メソッドの追加**　ResultSet.findColumn と ResultSet.get*Type* メソッドがサポートされるようになりました。
- **Ultra Light Java からの情報へのアクセス**　**JdbcConnection.getLastIdentity** メソッド、**getLastDownLoadTime** メソッド、**JdbcDatabase.countUploadRows** メソッドを使用すると、有用な情報にアクセスできます。これらの機能は、以前は C/C++ アプリケーションのみで使用可能でした。
- **Ultra Light Java でのユーザ認証**　Java バージョンの Ultra Light で、ユーザ認証がサポートされるようになりました。
- **HotSync 同期の進捗状況の表示**　デスクトップ・コンピュータの [HotSync Progress] ダイアログには、Ultra Light アプリケーションとの同期の進捗状況が表示されます。
- **HotSync の設定**　Palm Desktop から HotSync conduit を設定できます。
- **Ultra Light アプリケーションからのスクリプトの自動生成**　Ultra Light アプリケーションでは、同期スクリプトを自動的に生成できるように、Mobile Link サーバへのカラム名を用意するようになりました。
- **C++ API からカラムの SQL データ型を取得**　GetColumnSQLType メソッドは、カラムのデータ型を返します。
- **同期中のチェックポイント (オプション)**　大量の更新をダウンロードする同期は、Ultra Light データベースのサイズが大幅に増加する原因となる場合があります。この増加は、同期中にチェックポイントを実行することで制限できます。新しく **checkpoint_store** 同期パラメータを指定して、チェックポイントを制御します。デフォルトでは、チェックポイントは実行されません。

  詳細については、「Checkpoint Store 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。

バージョン 8.0.1 での動作の変更

この項では、SQL Anywhere Studio バージョン 8.0.1 のコンポーネントに導入された動作の変更について説明します。

Adaptive Server Anywhere の動作の変更

ここでは、これまでのバージョンとは異なる動作をリストにして説明します。

- **名前を変更するトランザクション・ログ・ファイルの新しい命名規則**　バックアップ中に変更されるトランザクション・ログ・ファイルの最後の 2 桁の数字は、2 つの文字に変更されました。たとえば、最初のバックアップが実行された日付が 2000 年 12 月 10 日の場合、ログ・ファイル名は *00121001.log* ではなく *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。この方法を使用すると、1 日に可能なバックアップ数が 100 から 676 に増えます。
- **LOAD TABLE による計算カラムの再計算**　LOAD TABLE は、計算カラムを検出し、テーブルに挿入されている各ローに対して計算カラムを評価するようになりました。
- **Adaptive Server Anywhere コンソール・ユーティリティ (dbconsole) による再接続**　以前の Adaptive Server Anywhere コンソール・ユーティリティ (dbconsole) セッションでは 1 つの接続のみが可能でした。接続は、アプリケーションを終了することなく、切断と再接続ができるようになりました。

廃止されサポートされなくなった機能

このリストには、サポートを終了した機能の中で既存のアプリケーションに影響のあるものがまとめられています。

- **使用されなくなった DEBUG 接続パラメータ**　DEBUG 接続パラメータが使用されなくなりました。LOG パラメータを使用すると、デバッグ情報を含むログ・ファイルを作成できます。バージョン 8.0.1 からは、DEBUG=YES;LOG=filename が行ってきたことを LOG=filename が行います。

  詳細については、「接続パラメータ」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **使用されなくなった AGENT 接続パラメータ**　AGENT 接続パラメータが使用されなくなりました。CommLinks パラメータを適切なプロトコル・オプションとともに使用すると、AGENT パラメータと同じ動作になります。

  詳細については、「接続パラメータ」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **Port 接続プロパティが削除**　port 接続パラメータが削除されました。

- **Adaptive Server Anywhere 変換ドライバが削除**　変換ドライバは使用しないことをおすすめします。文字セット変換は、サーバが自動的に処理します。
- **SharedMemory を最初に試行**　LINKS= 接続パラメータで指定したポートは、指定した順序で接続が試みられていました。今後は、sharedmemory (shmem) ポートが指定されている場合は、最初にこのポートへの接続が試みられ、その後は指定されている順序で接続が試みられるようになりました。
- **グローバル・オートインクリメント**　デフォルト値が 0 から 2147483647 に変更されました。global_database_id を 0 に設定し、1 から始まる値を生成できるようになりました。

Mobile Link の動作の変更

- **タイムスタンプ不一致の通知**　統合データベースとリモート・データベースのタイムスタンプが異なる場合、Mobile Link サーバによって、同期ごとの警告がログに記録されます。
- **グローバル・オートインクリメント**　デフォルト値が 0 から 2147483647 に変更されました。global_database_id を 0 に設定し、1 から始まる値を生成できるようになりました。

  global_database_id が設定されていないか、デフォルト値に設定されている場合、グローバル・オートインクリメント値を生成しようとすると、結果は NULL になります。この値を NULL 入力不可のプライマリ・キー・カラムに挿入しようとすると、通常はエラーが発生します。これは、global_database_id オプションが設定されていないことを示します。

  global_database_id に 0 を設定できないようにすると、1 から始まる値ではなく、カラムに指定された分割サイズから始まる値が生成されます。

  詳細については、「global_database_id オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **dbmlstop によるソフト・シャットダウン**　デフォルト (-w、-f、-h、-t のいずれも設定されていない場合) では、dbmlstop によってソフト・シャットダウンが行われます。これは、現在の同期が完了すると、新しい接続が受け入れられずに停止することを意味します。

詳細については、「Mobile Link 停止ユーティリティ [mlstop]」『Mobile Link - サーバ管理』を参照してください。

Ultra Light の動作の変更

- **Palm データベースのバックアップ**　以前のリリースでは、ULUtil アプリケーションを使用してデータベースをバックアップすると、その後 HotSync 操作を実行するたびにデータベースがバックアップされました。

  ほとんどの Ultra Light データは、同期によって効果的にバックアップされます。明示的なバックアップの最も一般的な用途は、配備用の初期データベースの作成であるため、ほとんどの場合、HotSync での継続的なバックアップは不要な動作です。現在のバージョンでは、Ultra

Light アプリケーションが起動するたびに、その後の HotSync 操作でのバックアップが無効になります。

HotSync が実行されるたびにデータベースのバックアップを明示的に要求する場合は、UL_STORE_PARMS マクロで **palm_all_backup** パラメータを設定します。

廃止され、サポートされなくなった機能

Ultra Light による、ScoutSync テクノロジを使用した Palm Computing Platform での同期サポートは終了しました。バージョン 8.0.x では、ScoutSync バージョン 3.6 までが引き続きサポートされますが、SQL Anywhere Studio の今後のメジャー・リリースからは ScoutSync はサポートされません。

第 8 章

バージョン 8.0.0 の新機能

目次

バージョン 8 の新機能

この項では、SQL Anywhere Studio バージョン 8 に導入された新機能について説明します。

Adaptive Server Anywhere の新機能

この項では、Adaptive Server Anywhere バージョン 8.0 の新機能について説明します。新機能について主要なものからそうでないものまですべてを示し、各機能の詳細についてのマニュアル内の参照先も記述しています。

このマニュアルの製本版を使用していて、SQL Anywhere Studio のマニュアル・セット全巻をお持ちでない場合、各機能の詳細についてはオンライン・マニュアルを参照してください。

一部の新機能では、データベースをバージョン 8 にアップグレードするか、データベースをアンロードして再ロードすることでデータベースのファイル・フォーマットをアップグレードする必要があります。特定の機能にアクセスするためにデータベースまたはファイル・フォーマットのアップグレードが必要な場合は、稼働条件が下記に記載されています。

これらのタスクを実行する方法については、「SQL Anywhere 10 へのアップグレード」 349 ページを参照してください。

Adaptive Server Anywhere の新機能は、次の見出しの下にグループ分けされています。

クエリ処理とデータベース・パフォーマンス

- **クエリ処理の向上**　このバージョンでは、クエリ実行エンジンとオプティマイザが強化されています。これによって、パフォーマンス、特に複雑なクエリのパフォーマンスが大幅に向上しています。Adaptive Server Anywhere のクエリ処理は、次の点が強化されています。
  - ジョインの内部処理の高度化。

- 代替アクセス・プランへのアクセスに使用されるオプティマイザのコスト・モデルの向上。
- 実行モデルの向上。

  変更の大半は、内部の変更です。詳細な説明については、「クエリの最適化と実行」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

これらの変更により、これまでのように結果の実体化が必ずしも非効率的とは言えなくなります。テンポラリ・ワーク・テーブルを使用することによって、クエリを非常に効率的に実行できる場合もあります。詳細については、「クエリ処理におけるワーク・テーブルの使用 (all-rows 最適化ゴールの使用)」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

オプティマイザは、コストに基づいてインデックスを選択するようになり、これまでのリリースのように述語の選択性だけに頼ることはなくなりました。

強化されたクエリ処理のほとんどは、データベースのアップグレードを必要としません。このリリースより前に作成されたデータベースで新しいコスト・モデルを使用するには、データベースをアンロードして再ロードすることによってそのデータベースのファイル・フォーマットをアップグレードする必要があります。

- **新しいインデックス・タイプ**　複数のカラム・インデックスと幅の広いカラムを含むインデックスについて、パフォーマンスを向上させる新しいタイプのインデックスが追加されました。新しいインデックスは、圧縮 B ツリー・インデックスといいます。

  Adaptive Server Anywhere は、インデックスの幅 (インデックス内の全カラムの幅の合計) に基づいて、適切なタイプのインデックスを自動的に作成します。圧縮 B ツリー・インデックスは、インデックスの幅が 10 バイト以上で、ページ・サイズの 1/8 未満から最大 256 バイトまでのときに作成されます。それ以外の場合は、ハッシュ B ツリー・インデックスが作成されます。

  CREATE INDEX 文の WITH HASH SIZE 句は、廃止されました。

  このリリースより前に作成されたデータベースで新しいタイプのインデックスを使用するには、データベースをアンロードして再ロードすることによってそのデータベースのファイル・フォーマットをアップグレードする必要があります。

  新しい制約が追加されています。外部キー・インデックスのサイズとタイプは、対応するプライマリ・キー・インデックスと同じである必要があります。

  dbunload は、デフォルト (WITH HASH SIZE 10) で指定された場合は、ハッシュ・サイズ指定を省略するようになりました。

- **新しいデータベース・オプション optimization_goal**　クエリ処理の最適化の対象を、最初のローを迅速に返すこと、または完全な結果セットを返すコストを最小限に抑えることのどちらかに指定します。デフォルトは、最初のローの最適化です。

  詳細については、「optimization_goal オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **テーブル・スキャンのパフォーマンスの強化**　Adaptive Server Anywhere 8.0 で作成された 2 K、4 K、または 8 K ページのデータベースでは、テーブルの逐次スキャンを必要とするクエリのパフォーマンスが強化されました。大きなテーブルに対しては、ページ・マップとも呼ばれるビットマップが Adaptive Server Anywhere によって作成されます。特定のテーブルのデータを含むすべてのページがビットマップにリストされます。この機能によって、1 回の I/O 操作だけで大きなテーブルを検索できます。

  詳細については、「テーブルとページのサイズ」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

  このリリースより前に作成されたデータベースでこの拡張機能を利用するには、データベースをアンロードして再ロードすることによってそのデータベースのファイル・フォーマットをアップグレードする必要があります。

- **チェックポイント・ログの記憶領域の向上**　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る連続ページ内に格納されるようになりました。これによって、チェックポイント・ログ内の項目に対して逐次スキャンと複数ページの書き込みが可能になり、パフォーマンスが向上しました。

  チェックポイント・ログの詳細については、「チェックポイントとチェックポイント・ログ」『SQL Anywhere サーバ - データベース管理』を参照してください。

  このリリースより前に作成されたデータベースでこの拡張機能を利用するには、データベースをアンロードして再ロードすることによってそのデータベースのファイル・フォーマットをアップグレードする必要があります。

- **プランのキャッシュ**　Adaptive Server Anywhere では、クエリと、ストアド・プロシージャ、ユーザ定義関数、トリガで実行される INSERT 文、UPDATE 文、DELETE 文の実行プランがキャッシュされます。キャッシュできるプランの最大数は、max_plans_cached オプションで指定します。プランのキャッシュを無効にするには、このオプションを 0 に設定します。

  詳細については、「プランのキャッシュ」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **デフォルトの I/O コスト・モデルの上書き**　ALTER DATABASE 文に CALIBRATE 句を指定すると、デフォルトの I/O コスト・モデルを上書きできるようになりました。

  詳細については、「ALTER DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **新しいデータベース・オプション max_plans_cached**　キャッシュに格納される実行プランの最大数を設定します。

  詳細については、「max_plans_cached オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しいデータベース・オプション min_table_size_for_histogram**　このオプションは、ヒストグラムを作成するテーブルの最小サイズを設定します。ヒストグラムには、1 つのカラムでの値の分散状況に関する情報が格納されます。オプティマイザはこの情報をもとに、効率的な実行プランを選択します。

セキュリティ

- **TCP/IP に対する強力な暗号化**　Adaptive Server Anywhere は、Solaris、Linux、NetWare、サポートされるすべての Windows OS (WindowsCE を除く) の TCP/IP ポートで証明書による暗号化をサポートするようになりました。強力な暗号化によって、クライアントとサーバ間で交換されるネットワーク・パケットの機密性と整合性が保護されます。この暗号化は、トランスポート・レイヤ・セキュリティ (TLS: Transport Layer Security) とも呼ばれています。

  ユーザは、Adaptive Server Anywhere の前のバージョンで使用されていた -e コマンド・ライン・オプションの代わりに、データベース・サーバの -ec コマンド・ライン・オプションを使用して、サーバの接続パラメータを設定できます。クライアントの接続パラメータは、暗号化接続パラメータと一緒に設定できます。

  詳細については、「-ec サーバ・オプション」『SQL Anywhere サーバ - データベース管理』と「Encryption 接続パラメータ [ENC]」『SQL Anywhere サーバ - データベース管理』を参照してください。

  この機能を使用するには、クライアントとサーバの両方でバージョン 8 ソフトウェアを使用する必要があります。データベースをアップグレードする必要はありません。

- **データベース・ファイルの強力な暗号化**　盗難の危険性が高いノートブック・コンピュータやラップトップ・コンピュータのセキュリティを強化するために、データベース・ファイル自体を強力に暗号化できるようになりました。

  詳細については、次を参照してください。

  - 「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバ - データベース管理』
  - 「-ek データベース・オプション」『SQL Anywhere サーバ - データベース管理』
  - 「-ep サーバ・オプション」『SQL Anywhere サーバ - データベース管理』
  - 「CREATE DECRYPTED FILE 文」『SQL Anywhere サーバ - SQL リファレンス』.

    暗号化データベース・ファイルを作成するには、バージョン 8 ソフトウェアを使用する必要があります。

SQL の機能

- **全外部ジョイン**　全外部ジョインがサポートされるようになりました。さらに、キーワード OUTER が、右、左、全外部ジョインのためのオプションになりました。

  詳細については、「外部ジョイン」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **CASE 文**　ANSI 規格では、2 つの形式の CASE 文を使用できます。Adaptive Server Anywhere 8.0 は、両方の構文をサポートします。

詳細については、「CASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **WAITFOR 文** 指定された時間の間、または指定の時間になるまで現在の接続処理を遅らせます。

  詳細については、「WAITFOR 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **RAISERROR 文によって接続を拒否する** この文は、接続の拒否または制限に使用できるようになりました。

  詳細については、「RAISERROR 文 [T-SQL]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **タイム・ゾーンの調整** タイム・ゾーン間の日付／時刻値の調整を簡単に行えるように、次の新機能が追加されています。

  - **CURRENT UTC TIMESTAMP** サーバのタイム・ゾーン調整値を使用することでタイム・ゾーン値を調整します。
  - **DEFAULT UTC TIMESTAMP** INSERT にデフォルト値を指定し、更新されたカラムをその値に設定します。
  - **TimeZoneAdjustment プロパティ** 新しい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返します。
  - **time_zone_adjustment オプション** 1 つの接続でタイム・ゾーンの調整を修正できます。

- **新しい照合関数** SORTKEY 関数は、文字データのソートに使用できる値を生成します。SORTKEY 関数を使用すれば、Adaptive Server Anywhere のデフォルトの照合よりも細かいソートを実行できます。

  COMPARE 関数を使用すれば、代替照合規則に基づいて 2 つの文字列を直接比較できます。

  詳細については、「SORTKEY 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』と「COMPARE 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **ERRORMSG 関数** 新しい SQL 関数 ERRORMSG を使用して、エラー・メッセージを取得できます。

  詳細については、「ERRORMSG 関数 [その他]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **データ型変換関数** ISDATE 関数と ISNUMERIC 関数は、文字列をそれぞれ日付または数値に変換できるかどうかをテストします。

  詳細については、「ISDATE 関数 [データ型変換]」『SQL Anywhere サーバ - SQL リファレンス』と「ISNUMERIC 関数 [その他]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

開発ツールと管理ツール

- **アクセシビリティ機能** SQL Anywhere Studio は、米国連邦リハビリテーション法 508 条に準拠しています。ユーザ・インタフェースとマニュアルはこの法律の規定に従って作成されています。このソフトウェアには、アクセシビリティ・ツールの使用を可能にするアクセシビリティ有効化コンポーネントが含まれています。アクセシビリティ有効化コンポーネントは、デフォルトではインストールされません。

  詳細については、「SQL Anywhere 10 のコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

- **クエリ・エディタ** Interactive SQL にグラフィカルなクエリ・エディタが追加されました。クエリ・エディタを使用すると、SQL コードを使用せずに SELECT 文を作成または編集できます。クエリ・エディタを開くには、Interactive SQL で [ツール] – [クエリの編集] をクリックします。

  詳細については、「クエリ・エディタの概要」 『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

- **Interactive SQL と Sybase Central でデータ編集が可能** Interactive SQL 結果セットを編集するか、Sybase Central でテーブルとビューを編集することによって、データベースを更新できます。ローの値をコピー、編集、挿入、削除できます。

  Sybase Central に表示されるデータを、クリップボードにコピーできます。

  詳細については、「Interactive SQL での結果セットの編集」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Interactive SQL での SQL エスケープ構文処理のサポート** Interactive SQL では、JDBC ドライバで実装されている関数のライブラリへのアクセスを可能にする JDBC エスケープ構文をサポートするようになりました。

  詳細については、「JDBC エスケープ構文の使用」 『SQL Anywhere サーバ - プログラミング』を参照してください。

- **プロシージャのプロファイル表示** Sybase Central の [プロファイル] タブでは、ストアド・プロシージャ、関数、イベント、トリガの呼び出し回数と実行回数についての情報が表示されます。また、プロシージャの各行の実行速度についての情報も参照できます。プロファイリング情報は、Sybase Central と SQL のストアド・プロシージャを使用して入手できます。

  Sybase Central でプロシージャのプロファイリング情報を参照する方法については、「システム・プロシージャを使用したプロシージャ・プロファイリング」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

  SQL ストアド・プロシージャを使用してプロシージャのプロファイリング情報を取得する方法については、「sa_procedure_profile_summary システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』および「sa_procedure_profile システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

この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。

- **アクセス・プラン情報表示の向上** アクセス・プランの表示方法に、グラフィカル表示と統計付きグラフィカル表示という 2 つの新しい方法が追加されました。これらの新しい方法ではクエリの処理コストに関する情報がさらに詳細に表示されるので、クエリのサブセットのコストを調べることができます。デフォルトのアクセス・プランは、グラフィカルなプランに設定されるようになりました。長いプランと短いプランは、Adaptive Server Enterprise で使用される Ariadne 構文に基づいて作成され、新しい省略形が追加されています。

  詳細については、「実行プランの解釈」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **[結果] ウィンドウ枠でのクエリ実行プランの表示** Interactive SQL の [結果] ウィンドウ枠に、[結果] タブが追加されました。[結果] タブにはクエリの結果が、[プラン] タブにはクエリの実行プランが表示されます。以前は、クエリ実行プランは Interactive SQL の [メッセージ] ウィンドウ枠に表示されていました。

  詳細については、「[オプション] ダイアログ：[結果] タブ」『SQL Anywhere 10 - コンテキスト別ヘルプ』を参照してください。

- **[結果] ウィンドウ枠での Ultra Light プランの表示** Interactive SQL の [結果] ウィンドウ枠に、[Ultra Light プラン] タブが追加されました。このタブには、Ultra Light プランの最適化方法が XML フォーマットの文字列で表示されます。

- **OUTPUT 文を使用した XML のエクスポート** クエリ結果を XML 形式でエクスポートできます。出力には、埋め込み DTD が含まれます。バイナリ値は、2 桁の 16 進数文字列として表されるバイナリ・データとして CDATA ブロック内にエンコードされます。

  詳細については、「OUTPUT 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **Interactive SQL のバッチ・オプション** -codepage と -onerror コマンド・ライン・オプションを使用して、Interactive SQL でのバッチ・ファイルの実行を制御する機能が追加されました。また、-dl コマンド・ライン・オプションを使用すると、バッチ・ファイルのデバッグに役立つフィードバックも提供されます。

  詳細については、「Interactive SQL ユーティリティ (dbisql)」『SQL Anywhere サーバ - データベース管理』を参照してください。

アプリケーション開発

- **新しいカーソル・タイプ** Adaptive Server Anywhere のカーソルが強化されて、整理されたセマンティックを提供したり、キーセット駆動型カーソルなどの新しいタイプのカーソルに適合するようになり、また、新しいクエリ最適化機能を最大限に利用できるようになりました。

  詳細については、「SQL Anywhere のカーソル」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **長いカラムのフェッチの向上**　1 回の操作でフェッチできるデータ量が 32 KB から、設定可能な値までに増えました。デフォルトは 256 KB です。ODBC では、SQL_ATTR_MAX_LENGTH 文属性を使用してこの値を設定できます。Embedded SQL では、DT_LONGVARCHAR 型と DT_LONGBINARY 型を使用します。

  詳細については、「データの取り出し」『SQL Anywhere サーバ - プログラミング』と「長い値の送信と取り出し」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **データベースのプロパティを取得する新しい Embedded SQL 関数**　関数 **db_get_property** を使用すると、データベースのプロパティを取得できます。

  詳細については、「db_get_property 関数」『SQL Anywhere サーバ - プログラミング』を参照してください。データベース・プロパティの詳細については、「データベース・プロパティの概要」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **blocking_timeout option**　新しい blocking_timeout オプションを使用すると、ロックを取得するまでにトランザクションがどの程度の期間待つかを制御できます。

  詳細については、「blocking_timeout オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **return_date_time_as_string オプション**　return_date_time_as_string オプションでは、jConnect と Open Client を介して、日付、時刻、タイムスタンプの値がどのように戻されるかを制御できます。

  詳細については、「return_date_time_as_string オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

管理機能とトラブルシューティング

バージョン 8 では、上記の Sybase Central に対する管理機能の強化のほかに、次の管理機能も強化されています。

- **アクセスを中断させない場合のテーブル・パフォーマンスの向上**　データベースへのアクセスを中断させないという稼働条件のためにデータベース全体の再構築が不可能なときは、REORGANIZE TABLE 文を使用してパフォーマンスを向上させることができます。この文を使って、テーブル内のローの断片化を解除したり、DELETE によって散在したインデックスを圧縮したりします。また、インデックス・ツリーに含まれるレベル数を減らすほかに、テーブルとそのインデックスを格納するページの合計数も減らします。

  プライマリ・キー、外部キー、またはインデックスに基づいてテーブルを再編成するには、データベースを Adaptive Server Anywhere バージョン 7 以上にしてください。

  詳細については、「REORGANIZE TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **高速データベース検証**　データベースの検証にかかる時間を短縮する新しいタイプの検証チェックが追加されました。このオプションは、小さなサイズのキャッシュを使用して大きなデー

タベースを検証する場合に、特に有用です。影響を受けるツールには、sa_validate システム・プロシージャ、検証ユーティリティ (**dbvalid**)、VALIDATE TABLE 文などがあります。

詳細については、「データベース検証時のパフォーマンスの改善」『SQL Anywhere サーバ - データベース管理』を参照してください。

このリリースより前に作成されたデータベースでこの機能を使用するには、データベースをアンロードして再ロードすることによってそのデータベースのファイル・フォーマットをアップグレードする必要があります。

- **バックアップで未処理トランザクションの完了を待機する必要なし** バックアップ命令でトランザクション・ログをトランケートするか名前を変更する必要がある場合、コミットされていないトランザクションは新しいトランザクション・ログに持ち越されます。つまり、サーバがバックアップを開始する前に、未処理のトランザクションをコミットまたはロールバックする必要はありません。

  詳細については、「ログ変換ユーティリティ (dbtran)」『SQL Anywhere サーバ - データベース管理』と「バックアップの内部メカニズム」『SQL Anywhere サーバ - データベース管理』を参照してください。

  このリリースより前に作成されたデータベースでこの機能を使用するには、データベースをアンロードして再ロードすることによってそのデータベースのファイル・フォーマットをアップグレードする必要があります。

- **断片化統計の取得** ファイル、テーブル、インデックスの断片化は、どれもパフォーマンスを低下させる可能性があります。Adaptive Server Anywhere 8.0 では、Windows NT 上でデータベースを起動すると、各 DB 領域でのファイルのフラグメント数に関する情報が自動的にサーバに表示されます。

  データベース管理者は、新しいシステム・プロシージャ sa_table_fragmentation と sa_index_density を使用して、データベースのテーブルとインデックスで発生した断片化に関する情報を取得できます。

  ファイルの断片化の詳細については、「ファイルの断片化の削減」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

  テーブルの断片化の詳細については、「テーブルの断片化削減」『SQL Anywhere サーバ - SQL の使用法』と「sa_table_fragmentation システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  インデックスの断片化の詳細については、「インデックスの断片化削減」『SQL Anywhere サーバ - SQL の使用法』と「sa_index_density システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **接続のために準備された最新の SQL 文を取得する** データベース・サーバの -zl コマンド・ライン・オプションを有効にすると、サーバ上のデータベースへのそれぞれの接続に対して準備された最新の SQL 文を取得できます。この機能は、remember_last_statement 設定で sa_server_option ストアド・プロシージャを使用して有効にすることもできます。

この機能が有効になっている場合、**LastStatement** プロパティ関数はサーバ上のデータベースへの現在の接続用に準備された最新の SQL 文を返し、sa_conn_activity システム・プロシージャはすべての接続用に準備された最新の SQL 文を返します。

詳細については、「-zl サーバ・オプション」『SQL Anywhere サーバ - データベース管理』、「sa_conn_activity システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』、および「sa_server_option システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **-cw コマンド・ライン・オプション** このサーバ・オプションを使用すると、Windows 2000、Windows XP、Windows Server 2003 で最大 64 GB のキャッシュ・サイズを使用できます。

  詳細については、「-cw サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **-qp オプション** このサーバ・オプションを使用すると、サーバ・メッセージ・ウィンドウにパフォーマンスに関するメッセージが表示されないようにできます。

  詳細については、「-qp サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **サーバ・ログのデバッグ機能の向上** 接続デバッガにロギングされる情報が改善されて、接続試行中の部分に関するより詳細なコンテキストが提供され、CONN: プレフィクスが削除され、TCP/IP メッセージ数が増加されました。

- **データベースが保持できるプロシージャが増加** システム・テーブル SYSPROCEDURE、SYSPROCPARM、SYSPROCPERM、SYSTRIGGER のプライマリ・キーが、SMALLINT から UNSIGNED INT に変更されました。この変更によって、データベースが保持できるプロシージャ数が増加しています。

  データベースが保持できるプロシージャ数の詳細については、「SQL Anywhere のサイズと数の制限」『SQL Anywhere サーバ - データベース管理』を参照してください。

  この機能を使用するには、データベースのファイル・フォーマットをアップグレードする必要があります。

- **クエリ・パフォーマンスのモニタリング** クエリ・パフォーマンスを測定する新しいシステム・プロシージャとユーティリティが追加されました。

  詳細については、「sa_get_request_profile システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』、「sa_get_request_times システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』、および「クエリのパフォーマンスのモニタ」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **新しい診断プロパティ** これらのプロパティを使用すると、接続、データベース、現在のデータベース・サーバについての情報を取得できます。今回のリリースには、次の接続プロパティが追加されています。

  - UtilCmdsPermitted プロパティ
  - TempTablePages プロパティ

- LastStatement プロパティ
- PacketSize プロパティ
- max_plans_cached プロパティ
- QueryCachePages プロパティ
- QueryLowMemoryStrategy プロパティ

  詳細については、「接続レベルのプロパティ」 『SQL Anywhere サーバ - データベース管理』を参照してください。

今回のリリースには、次のデータベース・プロパティが追加されています。

- DBFileFragments プロパティ
- LogFileFragments プロパティ
- BlobArenas プロパティ
- SeparateForeignKeys プロパティ
- VariableHashSize プロパティ
- TableBitMaps プロパティ
- FreePageBitMaps プロパティ
- SeparateCheckpointLog プロパティ
- Histograms プロパティ
- LargeProcedureIDs プロパティ
- PreserveSource プロパティ
- TransactionsSpanLogs プロパティ
- Capabilities プロパティ
- TempTablePages プロパティ
- CompressedBTrees プロパティ
- ProcedurePages プロパティ
- QueryCachePages プロパティ
- QueryLowMemoryStrategy プロパティ

  詳細については、「データベース・レベルのプロパティ」 『SQL Anywhere サーバ - データベース管理』を参照してください。

今回のリリースには、次のサーバ・プロパティが追加されています。

- MachineName プロパティ

- IsJavaAvailable プロパティ
- PlatformVer プロパティ

  詳細については、「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **パフォーマンス・モニタ統計値の追加** 今回のリリースでは、複数のパフォーマンス・モニタ統計値が追加されています。

  詳細については、「パフォーマンス・モニタの統計値」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **ログイン・プロシージャによって接続を拒否する** login_procedure オプションを使用すると、新しい接続ごとにストアド・プロシージャを呼び出すようにできます。このプロシージャによって、データベース接続を拒否することもできます。

  詳細については、「login_procedure オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **dbsvc の強化** Windows サービスを管理するための dbsvc コマンド・ライン・ユーティリティが拡張され、システム・コマンドの **net start** と **net stop** を使ってサービスを開始または停止するときに使用するサービス名をリストしたり、他のサービスやグループへの依存性を処理したりすることができるようになりました。

  詳細については、「Windows 用サービス・ユーティリティ (dbsvc)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **ストアド・プロシージャのソース・フォーマットを保持** スペースや改行を含むソース・フォーマットがコメントとしてデータベースに格納されるようになりました。このコメントは、プロシージャ・プロファイリングに使用されます。

クライアント／サーバ接続

- **向上したバッファ・サイズのネゴシエーション** クライアントとサーバの両方で、バッファ・サイズを個別に指定できるようになりました。

  この機能を使用するには、クライアントとサーバの両方でバージョン 8 ソフトウェアを使用する必要があります。データベースをアップグレードする必要はありません。

- **通信の圧縮** 新しいタイプの通信の圧縮によって、無線ネットワーク、一部のモデム、シリアル・リンク、一部の WAN などの、帯域幅が限定されたネットワークでデータを転送する場合のパフォーマンスを向上させることができます。

  詳細については、「パフォーマンス改善のための通信圧縮設定の調整」『SQL Anywhere サーバ - データベース管理』を参照してください。

  この機能を使用するには、クライアントとサーバの両方でバージョン 8 ソフトウェアを使用する必要があります。データベースをアップグレードする必要はありません。

- **dbping の強化**　dbping ユーティリティに、接続問題の診断を支援するオプションが追加されました。これらの機能には、ODBC を使用して接続する機能と、接続時の接続、データベース、サーバのプロパティをレポートする機能が含まれています。

  詳細については、「Ping ユーティリティ (dbping)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **TDS のデバッグを抑制するオプション**　suppress_tds_debugging オプションは、TDS デバッグ情報をサーバ・メッセージ・ウィンドウに表示するかどうかを制御します。

  詳細については、「suppress_tds_debugging オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **PrefetchBuffer 接続パラメータ**　この接続パラメータを使うと、プリフェッチされたローを格納するメモリの最大容量を指定できます。

  詳細については、「PrefetchBuffer 接続パラメータ [PBUF]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **PrefetchRows 接続パラメータ**　PrefetchRows 接続パラメータを使うと、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指定できます。状況によっては、クライアントによってデータベース・サーバからプリフェッチされるローの数を増やすと、クエリのパフォーマンスが向上することがあります。

  詳細については、「PrefetchRows 接続パラメータ [PROWS]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **クライアントでアイドル・タイムアウトを指定可能**　IDLE 接続パラメータを使用すると、各クライアントでアイドル・タイムアウトを指定できます。これまでは、サーバへのすべての接続で、-ti サーバ・コマンド・ライン・オプションを使用して共通のアイドル・タイムアウトを指定していました。

  詳細については、「Idle 接続パラメータ」『SQL Anywhere サーバ - データベース管理』を参照してください。

データベース内の Java

データベース内の Java には、次の新機能が含まれています。

- **Java 2 のサポート**　データベース内の Java は、Java 2 (JDK 1.2 と 1.3) と Java のクラスを使用できるようになりました。

  この機能を使用するには、ALTER DATABASE を使用するか、dbupgrad コマンド・ライン・ユーティリティで -jdk オプションを指定してデータベースをアップグレードする必要があります。

- **JDBC 2.0**　データベース内の Java クラスでは、JDBC 2.0 インタフェースを使用してデータにアクセスできるようになりました。

この機能を使用するには、ALTER DATABASE を使用するか、dbupgrad コマンド・ライン・ユーティリティで -jdk オプションを指定してデータベースをアップグレードする必要があります。

- **診断プロシージャ** 新しいシステム・プロシージャ **sa_java_loaded_classes** は、Java 仮想マシンによってロードされたすべてのクラスをリストします。

  詳細については、「sa_java_loaded_classes システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  この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。

- **セキュリティ・マネージャ** 組み込みセキュリティ・マネージャを使用するか、ユーザ独自の実装で、セキュリティ関連の Java 機能に対するアクセスを制御できます。

  詳細については、「Java のセキュリティ管理」『SQL Anywhere サーバ - プログラミング』を参照してください。

マニュアル

Adaptive Server Anywhere マニュアル・セットに、情報の検索、アクセス、使用がより短時間で行える新機能がいくつか追加されました。

- **マニュアルの再構成** 前回のリリース以降、マニュアル・セットは、次の 2 点で大幅な変更が行われています。
  - 『*レプリケーションおよび同期ガイド*』は、2 冊に分割され、2 種類の同期テクノロジを別々に説明しています。新しいマニュアルは『*Mobile Link 同期ユーザーズ・ガイド*』と『*SQL Remote ユーザーズ・ガイド*』です。
  - Adaptive Server Anywhere の『*ユーザーズ・ガイド*』、『*プログラミング・インタフェース*』、『*リファレンス・マニュアル*』は、『*データベース管理ガイド*』、『*SQL ユーザーズ・ガイド*』、『*SQL リファレンス・マニュアル*』、『*プログラミング・ガイド*』に置き換えられました。データベースに関するエラー・メッセージは、適切なマニュアルに移動されました。新しい構成によって、各マニュアルは、製本版として扱いやすいサイズになりました。
- **新しいコンテキスト別ヘルプ** Sybase Central、Interactive SQL、Adaptive Server Anywhere デバッガ、クエリ・エディタなどのすべてのユーザ・インタフェース・ツールは、プラットフォームを問わない共通のコンテキスト別ヘルプ・システムを共有しています。このシステムは、オンライン・マニュアルと完全にリンクしています。
- **オンライン・マニュアルの強化** オンライン・マニュアルの HTML ヘルプ・バージョンに、SQL Anywhere の Web リンク、チュートリアル、プロシージャなどにすぐにアクセスできるメニュー・バーが追加されました。

その他

- **ハイバネーション後も持続する接続**　Embedded SQL、ODBC、または OLE DB クライアントからの接続は、コンピュータがハイバネーション状態にある間も持続されるようになりました。これまでは、同一コンピュータ上のクライアントとサーバ間の TCP/IP 接続は、指定された活性またはアイドル・タイムアウト時間より長くハイバネーション状態にあったコンピュータが復帰すると、切断されていました。

- **現在のライセンス情報の表示**　dblic ユーティリティでは、サーバを起動せずに、サーバの実行プログラムに関する現在のライセンス情報を表示できる引数を使用できるようになりました。

  詳細については、「サーバ・ライセンス取得ユーティリティ (dblic)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **カスタム照合の照合ラベルと照合名の表示**　dbinfo ユーティリティは、カスタム照合用の照合ラベルと照合名を返すようになりました。また、*dbtools.h* の a_db_info 構造体に collationnamebuffer と collationnamebufsize という 2 つのフィールドが追加されました。

- **sp_remote_tables システム・プロシージャ**　sp_remote_tables ストアド・プロシージャに、新しい引数 **tabletype** が追加されました。この引数は、リモート・テーブルのタイプを返します。

  引数 **tabletype** の詳細については、「sp_remote_tables システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **-ct コマンド・ライン・オプション**　-ct コマンド・ライン・オプションを使用すると、文字セット変換を有効または無効に設定できます。文字セット変換はデフォルトで有効に設定されるようになりました。変換を無効に設定するには、-ct- を指定します。文字セット変換を有効にするには、-ct+ と指定します。

- **リモート・テーブルの外部キー情報の取得**　2 つの新しいストアド・プロシージャ sp_remote_exported_keys と sp_remote_imported_keys を使用して、リモート・テーブルの外部キーと対応するプライマリ・キーに関する情報を取得できるようになりました。

  詳細については、「sp_remote_exported_keys システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』と「sp_remote_imported_keys システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **xp_sendmail**　SMTP や MAPI を経由して電子メールを送信するための拡張ストアド・プロシージャが追加されました。詳細については、「xp_startsmtp システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』と「xp_stopsmtp システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  xp_sendmail ストアド・プロシージャには、任意の長さのメッセージを指定できます。このプロシージャに対する LONG VARCHAR パラメータの長さは、システムで使用できるメモリ容量によって制限されます。

  詳細については、「xp_sendmail システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **Log Transfer Manager のための Replication Server 12 の機能** LTM 設定ファイル内の **qualify_table_owners** パラメータは、プライマリ・データベースとレプリケーション・データベースで、異なるテーブル名、所有者、カラム名を使用できる Replication Server 12 の機能をサポートします。

  詳細については、「LTM 設定ファイル」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **ASANYSH8 環境変数** 新しい環境変数 ASANYSH8 が追加されました。Interactive SQL、Sybase Central、Adaptive Server Anywhere コンソール・ユーティリティ、デバッガでは、この環境変数を使用して、共有コンポーネントのディレクトリを検索します。

Mobile Link の新機能

バージョン 8.0 で導入されたソフトウェアの変更と追加を次に示します。

柔軟性

- **Java 同期論理** SQL 言語の代わりまたはそれに加えて、Java で記述した同期スクリプトを実装できるようになりました。これらのスクリプトは、Mobile Link Java 環境を使用して外部 JRE の中で実行できます。

  詳細については、「Java による同期スクリプトの作成」『Mobile Link - サーバ管理』を参照してください。

- **パブリケーションを使用した同期** Mobile Link クライアント内のすべてのデータを、同時に同期させる必要はなくなりました。その代わりに、データをパブリケーションに編成し、パブリケーションごとに別々に同期できます。パブリケーションと同期サブスクリプションには、以前の構文よりも簡単で精度の高い新しい構文が用意されています。

  詳細については、「SQL Anywhere クライアント」『Mobile Link - クライアント管理』を参照してください。

- **Mobile Link 同期を処理する Web サーバの設定** ファイアウォールの後ろ側で、Mobile Link サーバとの HTTP 同期を実行できるようになりました。標準的な Web サーバ用の Web サーバ・プラグインによって、Web サーバ経由の HTTP 同期を実行できます。

  詳細については、「リダイレクタによる Web サーバを経由した同期」『Mobile Link - サーバ管理』を参照してください。

- **Windows CE クライアントでの ActiveSync のサポート** Adaptive Server Anywhere と Ultra Light Windows CE Mobile Link のクライアントは、いずれも Windows CE ActiveSync 同期ソフトウェアを使用できます。

  詳細については、「ActiveSync 同期の使用」『Mobile Link - クライアント管理』を参照してください。

- **クライアントのコマンド・ライン機能の強化** CREATE/ALTER SYNCHRONIZATION SUBSCRIPTION 文とコマンド・ラインの両方で、拡張オプションを指定できます。

詳細については、「dbmlsync 構文」『Mobile Link - クライアント管理』を参照してください。

- **拡張オプションをデータベース内に格納できる** CREATE/ALTER SYNCHRONIZATION SUBSCRIPTION 文を使用して、拡張オプションと接続パラメータをデータベースに保存し、サブスクリプション、ユーザ、またはパブリケーションと関連付けることができます。*Dbmlsync* は、k の情報をデータベースから読み込みます。

  詳細については、「dbmlsync 構文」『Mobile Link - クライアント管理』を参照してください。

パフォーマンス

- **文ベースのアップロード** Mobile Link では、カーソルベースのアップロードよりも直感的に扱えるだけではなく処理速度も速い、文ベースのアップロードを行えるようになりました。文ベースのアップロードでは、**upload_insert**、**upload_delete**、**upload_update**、**upload_new_row_insert**、**upload_old_row_insert** の各イベントが使用されます。競合の解決には、**upload_fetch** スクリプトが使用されます。

  詳細については、「ローをアップロードするスクリプトの作成」『Mobile Link - サーバ管理』を参照してください。

- **マルチプロセッサの管理** 使用するプロセッサの最大数を設定するオプションが Mobile Link に新しく追加されました。-zt オプションを使用して、Mobile Link 同期サーバで使用されるリソースをより細かく制御できます。また、複数のプロセッサを使用した場合の ODBC の問題の検出や解決の際にも役立ちます。

  詳細については、「-zt オプション」『Mobile Link - サーバ管理』を参照してください。

- **ダウンロード確認のオプション** Mobile Link 同期クライアントは、ダウンロード確認なしで同期できるようになりました。したがって、Mobile Link サーバのワーカ・スレッドは、クライアントがダウンロードを適用するまで待つ必要がなく、早期にワーカ・スレッドを解放して次の同期を実行できます。ダウンロード確認は、オプションになりました。ダウンロード確認を省略することで、スループットが向上します。特に、処理の遅いクライアントでは顕著な効果が現れます。ダウンロード確認を省略した場合、統合データベース側では次の同期が開始されるまでダウンロードが成功したかどうかを知ることはできません。

  詳細については、「Mobile Link SQL Anywhere クライアントの拡張オプション」『Mobile Link - クライアント管理』を参照してください。

- **ダウンロード・ストリームのバッファ** Mobile Link サーバは、ダウンロード・キャッシュにダウンロード・ストリームをバッファするようになりました。ダウンロード・トランザクションをコミットするのにクライアントからの確認は必要ないので、バッファされたダウンロード・ストリームは、コミット後にクライアントに送信されます。これにより、これまでのようにネットワークの遅延によってダウンロード・トランザクションの処理が滞ることもなくなります。

  ダウンロード・ストリームは、Adaptive Server Anywhere クライアント側でもバッファできます。使用可能なバッファのサイズは、*dbmlsync* DownloadBufferSize 拡張オプションを使って設定できます。

詳細については、「Mobile Link SQL Anywhere クライアントの拡張オプション」『Mobile Link - クライアント管理』を参照してください。

- **接続スクリプトとテーブル・スクリプトのバルク・ロード** テーブルと version_id の特定のペアに対して初めて接続スクリプトまたはテーブル・スクリプトを要求すると、すべてのスクリプトがキャッシュにバルク・ロードされます。すべてのスクリプトを個別に取得するのではなく一括して取得することによって、パフォーマンスが向上します。
- **Mobile Link サーバのシャットダウン処理の強化** Mobile Link サーバが完全にシャットダウンを完了してから処理を実行するように、*dbmlstop* に通知することができます。また、*dbmlstop* を使用して、特定の Mobile Link サーバの名前を指定して停止できます。

  詳細については、「Mobile Link 停止ユーティリティ [mlstop]」『Mobile Link - サーバ管理』を参照してください。
- **接続タイムアウト** 指定された時間にわたって使用されなかった Mobile Link データベース接続は、サーバによって自動的に切断されるようになりました。タイムアウトは、-ct (接続タイムアウト) コマンド・ライン・オプションを使用して設定できます。

  詳細については、「-ct オプション」『Mobile Link - サーバ管理』を参照してください。
- **同時アップローダの最大数オプション** -wu コマンド・ライン・オプションを使用して、同時アップロードを許可するワーカ・スレッドの最大数を設定できます。この結果、一部の配備ではスループットが増大します。

  詳細については、「-wu オプション」『Mobile Link - サーバ管理』を参照してください。

セキュリティ

- **Mobile Link ユーザ認証** パスワードに基づくユーザ認証システムによって、Mobile Link のインストール環境のセキュリティがさらに強化されます。さらに、-zu を使用すると、authenticate_user スクリプトが定義されていない場合にユーザを自動的に追加できます。これによって、ユーザのスキーマ情報を Mobile Link の認証として使用できます。

  詳細については、「Mobile Link ユーザ」『Mobile Link - クライアント管理』を参照してください。
- **Mobile Link ユーザの管理** dbmluser ユーティリティが拡張され、システムからユーザを削除したり、システムにユーザを追加したりできるようになりました。このユーティリティについてはそのほかにも拡張機能が追加されています。dbmluser コマンド・ライン・オプション -pf、-pp、-pu は廃止され、それぞれ -f、-p、-u に置き換えられました。

  詳細については、「Mobile Link ユーザ認証ユーティリティ [mluser]」『Mobile Link - サーバ管理』を参照してください。

レポート機能の強化

- **統計スクリプト** Mobile Link に、同期統計を追跡するためのスクリプトが追加されました。これらの同期統計を収集すると、その後の同期のパフォーマンスの監視に使用できます。

詳細については、「synchronization_statistics 接続イベント」『Mobile Link - サーバ管理』、「synchronization_statistics テーブル・イベント」『Mobile Link - サーバ管理』、「upload_statistics 接続イベント」『Mobile Link - サーバ管理』、および「upload_statistics テーブル・イベント」『Mobile Link - サーバ管理』を参照してください。

- **詳細なネットワーク・エラー情報** Mobile Link のサーバとクライアントは、詳細なエラー情報を表示するようになり、発生したすべてのエラーを、エラー・コードを参照して解決できるようになりました。エラーをレポートしたネットワーク・レイヤ、実行されたネットワーク操作、エラー内容、システム固有のエラー・コードを確認できます。
- **エラー発生時に Mobile Link サーバに送信される Remote Adaptive Server Anywhere の出力ログ** 同期問題のトラブルシューティングは、リモート・ログと Mobile Link サーバ・ログの両方を検査するのが最も簡単な方法です。この新機能では、クライアント側でエラーが発生したときに、Adaptive Server Anywhere リモートの出力ログが Mobile Link サーバに送信されます。

  詳細については、「-e オプション」『Mobile Link - サーバ管理』を参照してください。
- **ログ・メッセージがワーカ・スレッドを識別する** Mobile Link サーバ・ログに表示されるメッセージは、そのメッセージをロギングしたワーカ・スレッドを示すようになりました。これによって、1 人のユーザが同時に同期させようとしたために発生したメッセージを識別できます。また、1 人のユーザが続けて 2 回同期させたときのメッセージも識別できます。
- **冗長ロギング** Mobile Link サーバの -v コマンド・ライン・オプションに追加の変更子を使用すると、Mobile Link サーバのロギングを設定できます。

  詳細については、「-v オプション」『Mobile Link - サーバ管理』を参照してください。
- **無視されたローがクライアントにレポートされる** スクリプトがないために、Mobile Link サーバがアップロードされたローを無視した場合、クライアントにメッセージが返されます。このメッセージは、Adaptive Server Anywhere クライアントからの警告として表示され、Ultra Light クライアントの ignored_rows 同期パラメータにも表示されます。

  詳細については、「Ignored Rows 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。

使いやすさ

- **最終ダウンロード・タイムスタンプ** 前回のダウンロードのタイムスタンプが Mobile Link クライアント・データベースに自動的に書き込まれます。
- **自動同期スクリプトの生成** スナップショットを使用した同期に適したスクリプトを生成するように、Mobile Link に指示できます。これらのスクリプトの作成とアクティブ化は、-za オプションで制御します。
- **サンプル同期スクリプトの生成** サンプル同期スクリプトを生成するように、Mobile Link に指示できます。-ze コマンド・ライン・オプションを使用して、サンプル・スクリプトを生成するかどうかを指定します。

適合性

- **一般的な RDBMS のサポート**　Mobile Link は、統合データベースとして、Oracle 8i と 9i、Microsoft SQL Server 7、Microsoft SQL Server 2000、IBM DB2 などをサポートします。

  詳細については、「Mobile Link でサポートされる ODBC ドライバ」『Mobile Link - サーバ管理』を参照してください。

- **TCP/IP ストリームでの活性検出**　Mobile Link の同期中に使用される TCP/IP ベースのストリームは、クライアントとサーバの両側で、活性チェックを有効にする **keep_alive** と呼ばれる新しいパラメータを利用できるようになりました。

  詳細については、「-x オプション」『Mobile Link - サーバ管理』を参照してください。

Ultra Light の新機能

Ultra Light 8.0 には、次の新機能が導入されています。

セキュリティ

- **ユーザの認証**　これまでのリリースでは、Ultra Light データベースには、アクセスを管理するためのユーザ認証メカニズムがありませんでした。今回のリリースには、ユーザ認証メカニズムが組み込まれています。多くのリレーショナル・データベース管理システムのユーザ ID とは異なり、Ultra Light のユーザ ID は、テーブルやその他のデータベース・オブジェクトの所有権を意味しません。

- **データベースの暗号化**　データベースを暗号化することで、データのセキュリティを強化することができます。以下の 2 つの方法があります。

  - **強力な暗号化**　強力な暗号化アルゴリズムを使用してデータベースを暗号化すると、最高レベルのセキュリティを得ることができます。このセキュリティには、パフォーマンスの低下が伴います。この暗号化は、キーを使用するものであり、AES 128 ビット・アルゴリズムを使用しています。

  - **データベースの難読化 (Obfuscation)**　データベースを読みにくくすることで、データのセキュリティを強化することができます。難読化しない場合、16 進数エディタなどのツールを使ってデータベース内のデータを表示できます。難読化は、データを表示しようとする程度の軽い不正アクセスは防げますが、強力な暗号化のような厳重な保護は行いません。難読化では、強力な暗号化に伴うパフォーマンスの低下は発生しません。

    詳細については、「Ultra Light でのセキュリティの考慮事項」『Ultra Light - データベース管理とリファレンス』を参照してください。

- **Ultra Light Java アプリケーションの安全な同期**　Certicom トランスポート・レイヤ・セキュリティを使用した安全な同期は、これまでは C/C++ の Ultra Light アプリケーションでのみ行えました。今回のリリースでは、Ultra Light Java アプリケーションでも行えるようになりました。

同期

- **ActiveSync 同期**　Windows CE デバイス上で動作する Ultra Light アプリケーションは、ActiveSync を使用して同期させることができます。

  詳細については、「アプリケーションへの ActiveSync 同期の追加」『Ultra Light - C/C++ プログラミング』を参照してください。

- **同期の柔軟性の向上**　同期させるデータをさらに効率よく柔軟に選択できるようにする新機能がいくつか追加されました。

  - パブリケーションを使用すると、データを複数のセットに分割し、セットごとに別々に同期させることができます。処理速度が遅い接続リンクなどでは、これによって時間が重要なデータを効率的に同期でき、時間が重要ではない他のデータは都合のよいときに同期できます。
  - ダウンロード専用同期を使用すると、読み込み専用テーブルを Ultra Light データベースに追加して、それらを効率的に同期できます。
  - テーブル内のデータが変更されているか否かに関係なく、毎回同期するようにテーブルをマークできます。この機能を使用すると、同期を制御する Ultra Light クライアント上のユーザ設定情報を管理できます。

- **グローバル・オートインクリメントのデフォルト・カラム値**　この機能を使って、同期中のデータベース内のプライマリ・キーの一意性を簡単に管理できます。

  詳細については、「オートインクリメント・カラムの分割サイズの上書き」『Mobile Link - クライアント管理』を参照してください。

- **Ultra Light ジェネレータの制御の追加**　ulgen と sqlpp 実行プログラム用のコマンド・ライン・オプションが追加されました。

  - **スクリプトのバージョン**　生成された同期スクリプトに、スクリプトのバージョンを関連付けることができます。
  - **ログ・クエリ実行プラン**　生成されたクエリのクエリ実行プランをエクスポートし、Interactive SQL で表示できます。

- **エラー・レポート**　**ul_synch_info** 構造の **stream_error** フィールドを使って、同期エラーの原因を判断できます。

  詳細については、「Stream Error 同期パラメータ」『Mobile Link - クライアント管理』を参照してください。

データベース管理

- **既存のデータベースの再使用**　Ultra Light のこれまでのリリースでは、データベース・アプリケーションが変更された場合は、データベースを再構築して同期させる必要がありました。今回のリリースより、データベースのスキーマが変更されないかぎり、新バージョンのアプリケーションで Ultra Light データベースを続けて使用できるようになりました。クエリを変更した場合は、そのクエリが新しいカラムを参照することで、生成済みのデータベースのスキーマが変更されないかぎり、新しいデータベースを必要としません。

- **データベースの断片化解除** Ultra Light の記憶領域は、空き領域を効率的に再使用するように設計されているので、通常の状況では明示的な断片化解除を必要としません。領域要件が極端に厳しいアプリケーションのために、明示的な断片化解除機能が用意されています。
- **ページ・サイズの選択** デフォルトの 4 KB ページに代わって 2 KB ページ・サイズを選択できます。

開発用の機能

- **CodeWarrior 7 のサポート** CodeWarrior 用の Ultra Light プラグインは、CodeWarrior バージョン 7 をサポートするようになりました。
- **eMbedded VIsual C++** このツールを使用した開発がサポートされ、CustDB サンプル・アプリケーション用の eMbedded Visual C++ プロジェクトが用意されました。
- **Palm OS 4.0 とファイル・ベースのデータ記憶装置** Ultra Light は Palm Computing Platform のバージョン 4.0 をサポートします。Palm ではバージョン 4.0 以降、さまざまな補助記憶装置スキームが導入されています。Palm 4.0 デバイスの拡張カードではファイル・ベースの Ultra Light データ・ストアを使用できます。

  詳細については、「ULEnableFileDB 関数 (旧式)」『Ultra Light - C/C++ プログラミング』を参照してください。
- **Palm Computing Platform の同期機能の向上** Palm Computing Platform の HotSync と ScoutSync による同期には、新しい簡略化された同期メカニズムが導入されています。新しい同期メカニズムには、これまでの同期メカニズムと比べて次のような長所があります。
  - 起動と終了が速い。
  - 同期の実行時に Palm デバイス上に追加の記憶装置が不要。
  - アプリケーションを複数回同期でき、そのたびに起動する必要がない。
  - ストリーム・パラメータの指定が不要。

  ULPalmDBStream 関数と **ULConduitStream** 関数は廃止されました。

  詳細については、「Palm アプリケーションに HotSync 同期を追加する」『Ultra Light - C/C++ プログラミング』を参照してください。
- **Palm Computing Platform での配備の簡略化** Ultra Light データベースの初期コピーをエンド・ユーザに配備できるので、最初の同期操作でユーザごとにデータの初期コピーをダウンロードする必要がありません。

  詳細については、「Palm アプリケーションの配備」『Ultra Light - C/C++ プログラミング』を参照してください。
- **Palm セグメントの処理方法の向上** Palm Computing Platform 用のアプリケーションを開発するときは、限定されたサイズのセグメントにアプリケーション・コードを分割する必要があります。

  このソフトウェアの以前のバージョンのセグメンテーション方法では、ユーザは Ultra Light で生成されるコードのセグメンテーションを制御できず、割り当てられるセグメントが多く

なりすぎる傾向がありました (これはパフォーマンスの低下を招きます)。新しいメカニズムでは、生成されるセグメントの数が減少するとともに、ユーザがセグメントの割り当てを制御できます。

- **Embedded SQL の LONG 値** DECL_LONGVARCHAR と DECL_LONGBINARY を使用して、LONG 値 (32 KB から 64 KB の間) にホスト変数を使用できます。
- **リファレンス・データベースでのアナライザのフック** Ultra Light ジェネレータは、分析処理の前後にストアド・プロシージャを呼び出すようになりました。
- **クエリ・プラン情報** Ultra Light ジェネレータで、Ultra Light アプリケーションでのクエリに使用されるアクセス・プランを出力できるようになりました。また、Interactive SQL から Ultra Light に使用されるアクセス・プランを表示することもできます。
- **スクリプトのバージョン制御** Ultra Light ジェネレータのコマンド・ラインで同期に使用されるスクリプト・バージョンを指定できます。
- **SQL と API 機能の追加** Ultra Light アプリケーションに対して、次の機能を使用できるようになりました。
  - **@@identity のサポート** Ultra Light で @@identity グローバル変数がサポートされるようになりました。この機能は、グローバル・オートインクリメントのデフォルト・カラム値に関連して使用できます。C++ API では、**ULConnection::GetLastIdentity()** メソッドを使用します。
  - **テーブル内のローの数** **ULTable::GetRowCount()** メソッドを使用すると、C++ API プログラミング・インタフェースからテーブル内のローの数を判断できます。Embedded SQL ユーザは、引き続き SELECT COUNT(*) FROM *table-name* 文を使用します。
  - **テーブル内のすべてのローの削除** **ULTable::DeleteAllRows()** メソッドを使用すると、C++ API プログラミング・インタフェースからテーブル内のすべてのローを削除できます。Embedded SQL ユーザは、引き続き DELETE FROM *table-name* 文を使用します。
  - **影響を受けたローの数** Eembedded SQL から SQLCOUNT マクロを使用して、最新の INSERT、UPDATE、または DELETE 文によって影響を受けたローの数を調べることができます。
  - **アップロードされるローの数** 同期する必要があるローの数を調べることができます。

    詳細については、「ULCountUploadRows 関数」『Ultra Light - C/C++ プログラミング』を参照してください。
  - **最終ダウンロード時間** Ultra Light アプリケーションから前回パブリケーションをダウンロードした時間を取得できます。

    詳細については、「ULGetLastDownloadTime 関数」『Ultra Light - C/C++ プログラミング』を参照してください。
  - **カーソル処理の追加** C++ API の **ULTable** クラスに、結果セット内のローを検索するためのメソッド (**FindFirst**、**FindNext**、**FindPrevious**、**FindLast**) が追加されました。

- **DUMMY システム・テーブルからのクエリ**　SELECT … FROM DUMMY 形式のクエリがサポートされるようになりました。
- **複数テーブルの更新**　複数のテーブルに対するカーソルによって、複数のテーブルを修正する更新を受け付けることができるようになりました。
- **Embedded SQL での LONG データ型の処理の向上**　DECL_LONGVARCHAR と DECL_LONGBINARY ホスト変数型を使用すると、1 回の操作で 32 KB を超えるデータを送信または検索ができます。

  詳細については、「Embedded SQL のデータ型」『Ultra Light - C/C++ プログラミング』を参照してください。

SQL Remote の新機能

- **イベント・フック・プロシージャ**　レプリケーション・プロセスのカスタマイズを可能にする一連のイベント・フック・プロシージャが追加されました。指定された名前のストアド・プロシージャを作成すれば、Message Agent がレプリケーションの実行中に行うアクションに対してさまざまな時点でカスタマイズを追加できます。

  詳細については、「SQL Remote イベント・フック・プロシージャ」『SQL Remote』を参照してください。

バージョン 8 での動作の変更

この項では、SQL Anywhere Studio バージョン 8 のコンポーネントに導入された動作の変更について説明します。

Adaptive Server Anywhere の動作の変更

ここでは、これまでのバージョンとは異なる動作について説明します。

新たに廃止され、サポートされなくなった機能のリストについては、「廃止されサポートされなくなった機能」 288 ページを参照してください。

- **データベース内の Java は、別途ライセンスが必要**　この結果、データベース作成時のデフォルト動作では、データベース内の Java のサポートが除外されます。

  Ultra Light ジェネレータが外部 Java 仮想マシンを使用するように変更されているので、データベース内の Java は、Ultra Light リファレンス・データベースで必要ではなくなりました。

  詳細については、「SQL Anywhere 10 のコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

- **集合関数と外部参照**　Adaptive Server Anywhere のバージョン 8 は、サブクエリでの集合関数の使用法を明確に規定した新しい SQL/99 標準に準拠しています。この変更は、Adaptive Server Anywhere の以前のバージョン用に記述された文の動作に影響を与えます。以前は正しかったクエリでエラー・メッセージが生成され、結果セットが変わる可能性があります。

  詳細については、「集合関数と外部参照」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **ユーザが提供する選択性推定**　Adaptive Server Anywhere では、アクセス・プランの選択を支援する選択性推定を明示的に指定できます。これらの推定は、ソフトウェアが選択したアクセス・プランが不適切な場合のパフォーマンス問題を回避するときに最も有効でした。新しい user_estimates 接続オプションは、ユーザが提供した選択性推定をオプティマイザが使用するか無視するかを制御します。

  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ている場合は、明示的な推定が不正確になり、オプティマイザが不適切なプランを選択する可能性があるので、user_estimates オプションの設定を OFF にすることをおすすめします。今回のバージョンでは、内部ジョイン・アルゴリズムなどのクエリ処理の強化が追加されているため、クエリのパフォーマンスが大幅に向上しています。

  ユーザが提供する選択性推定の詳細については、「user_estimates オプション [データベース]」 『SQL Anywhere サーバ - データベース管理』と「明示的な選択性推定」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **ローの順序**　バージョン 8.0 では、クエリ処理の向上による副作用として、ローの順序の確定性が低下しています。ORDER BY 句がない場合は、Adaptive Server Anywhere が最も効率のよい順序でローを返します。これは、結果セットの提示が、最後にアクセスしたローとその他

の要因によって変化する可能性があることを意味します。特定の順序でローが返されるようにする唯一の方法は ORDER BY を使用することです。

LIST 関数は、この変更によって特に影響を受ける関数の 1 つです。

- **アクセス・プランの変更**　Adaptive Server Anywhere の今回のリリースで選択されるアクセス・プランは、以前のリリースほどインデックスを使用しない傾向があります。テーブル・スキャンの効率を向上させ、アクセス・プランのコスト比較時に選択の幅があるコスト・モデルを使用することによって、インデックスの利便性をこれまでのバージョンよりも正確に評価できるようになりました。

- **カーソルの変更**　カーソルの強化の副効用として、このバージョンのカーソルは、定義された規格に以前よりも近い動作をするようになりました。このため、Adaptive Server Anywhere が ODBC やその他のインタフェースの要求に合致する動作を行うように、一部のカーソルの反応が変更される場合があります。たとえば、Embedded SQL の SCROLL カーソルではプリフェッチが実行できなくなったため、値の変更がカーソルに反映されます。

  この変更は、SQL_SUCCESS へのリターン・コードのみをチェックし、SQL_SUCCESS_WITH_INFO のチェックはしていない既存のアプリケーションに影響します。SQL_SUCCESS_WITH_INFO に対してチェックを行うアプリケーションでは、カーソルの動作が要求された動作と異なる場合は、警告を受け取ります。この警告は、SQLCODE=121、SQLSTATE 01S02 です。

  insensitive カーソルは更新できません。

  詳細については、「insensitive カーソル」 『SQL Anywhere サーバ - プログラミング』を参照してください。

- **ストアド・プロシージャの格納**　ストアド・プロシージャは、書かれたとおりに格納されるようになりました。Adaptive Server Anywhere は、ストアド・プロシージャの内部表現を作成し、これをプロファイリングに使用します。

- **挿入時の OPEN CURSOR はサポートされない**　INSERT 文でカーソルを開く機能は削除されました。更新可能なカーソルは、業界標準と同じように SELECT 文を使用して開くことができます。

- **ユーザ定義関数**　ユーザ定義関数のパラメータと戻り値は、キャッシュされるようになりました。SQL 文の中で関数が複数回使用される場合は、キャッシュされたパラメータ値は、関数を再評価するのではなく、使用されたキャッシュ済みの結果になります。以前のリリースでは、ユーザ定義関数が必要になるたびに再評価されていました。この新しい動作によってパフォーマンスが向上し、さらに一貫性のある結果が得られますが、以前のリリースのソフトウェアと比較すると結果が異なっている場合があります。

- **NUMBER(*) 関数の変更**　NUMBER 関数は、不確定な動作を回避するために使用を制限されています。NUMBER は、クエリの選択リストの中で、結果セットのローに連番を振ることを目的としていますが、この使い方はまだ許可されています。

  NUMBER 関数は、-1 の値を使用して絶対フェッチを実行した後にカーソルを逆向きに移動するなどの動作を行った場合に、以前のバージョンでは発生しなかった負の値を戻すことがあります。この新しい動作は、ISO/ANSI フェッチ・オフセットに対応しています。

NUMBER 関数を WHERE 句または HAVING 句など、多くの状況で使用すると、エラーが発生するようになりました。

詳細については、「NUMBER 関数 [その他]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **カスタム照合の変更** 以前は、照合ユーティリティの -d オプションは 3 つのパラメータを受け付けていましたが、現在使用できるパラメータは 2 つだけになりました。*cust-map-file* パラメータは、現在は受け付けられません。

  さらに、現在はスクリプト・ファイル *collsqmp.sql* と *custmap.sql* は存在しないため、それぞれ組み込み照合とカスタム照合で使用できません。

  新しく作成されたデータベースでは、SYSCOLLATIONMAPPINGS テーブルに照合マッピングを持つローが 1 つだけ格納されます。Adaptive Server Anywhere の以前のバージョンで作成されたデータベースでは、このテーブルに組み込み照合のためのローが格納されています。

- **トリガ名の変更** トリガ名は、データベース全体でユニークなものにする必要はなくなりました。ユニークである必要があるのは、トリガが適用されるテーブルの中だけです。したがって、DROP TRIGGER と COMMENT ON TRIGGER の構文が変更され、テーブルも指定した場合には所有者だけを指定できるようになりました。つまり、所有者だけでトリガを修飾する古いスクリプトでは、「テーブルが見つかりません」のエラーが発生します。

- **サンプル・データベース内のアドレスの変更** Adaptive Server Anywhere 9.0 のサンプル・データベース内のアドレスは、以前のリリースのものとは異なります。

- **内部 JDBC ドライバの JAR ファイル名の変更** 内部 JDBC ドライバのクラスは ASAJDBC ではなく ASAJRT という名前の JAR ファイルでインストールされるようになりました。

- **RESTORE DATABASE 文のパーミッション** RESTORE DATABASE 文を実行するのに、ユーティリティ・データベースに接続する必要はなくなりました。RESTORE DATABASE 文を実行するために必要なパーミッションは、-gu コマンド・ライン・オプションによって制御されます。

  詳細については、「RESTORE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **TDS 接続では空文字列を NULL 文字列で返す** tds_empty_string_is_null オプションは、TDS 接続でサーバが空文字列を返す場合、ブランク文字 1 文字を含む文字列または NULL 文字列のどちらで返すかを制御します。

  詳細については、「tds_empty_string_is_null オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **COMMENT 文の変更** 以前は、COMMENT ON INDEX の構文には、インデックスの所有者名がオプションで含まれていました。インデックス名に所有者とテーブルをオプションで入れることができるようになりました。COMMENT ON INDEX の構文は次のとおりです。

  COMMENT ON INDEX [[*owner*.]*table*.]*index-name* **IS** *comment*

詳細については、「COMMENT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **文字セット変換がデフォルトで有効** Adaptive Server Anywhere の以前のバージョンでは、文字セット変換はデフォルトで無効に設定され、有効にするには、-ct コマンド・ライン・オプションを指定する必要がありました。文字セット変換はデフォルトで有効になりましたが、-ct- コマンド・ライン・オプションを使用すると無効にできます。

  サーバが接続の文字セットとデータベースの文字セットが異なると判断すると、サーバはその接続との間で送受信されるすべての文字列に対して文字セット変換を適用します。

  サーバがデータベースと接続の文字セットが等しいと判断すると、サーバはその接続に対する文字セット変換を無効にします。

  ほとんどの場合、文字セット変換は有効に設定しておいてください。設定を無効にするのは、バイナリ・データがデータベースに挿入され、文字データとしてフェッチされた場合 (あるいはその逆の場合) です。サーバは文字データに対してのみ文字セット変換を適用するため、このような場合には、入力されたとおりのデータが返されないことがあります。この問題を回避するには、アプリケーションでバイナリ型を使って文字データの送信やフェッチを行わないようにします。

- **CONVERT、timestamp_format、date_format** timestamp_format または date_format オプションを使用するときに、大文字と小文字の混在する文字記号 (Mmm など) を指定すると、Adaptive Server Anywhere は使用されている言語に適切な大文字または小文字を選択します。また、CONVERT 関数は、使用されている言語に適切な大文字または小文字に文字日付を変換するようになりました。たとえば、英語では May、フランス語では mai が正しい大文字と小文字の用法です。

  詳細については、「date_format オプション [互換性]」『SQL Anywhere サーバ - データベース管理』、「timestamp_format オプション [互換性]」『SQL Anywhere サーバ - データベース管理』、「CONVERT 関数 [データ型変換]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **3 値的ブール論理への変更** 2 値的ブール論理は、*expr* = NULL の場合にのみ適用されます。ここで、*expr* はベース・カラムまたはベース・カラムについての式を指します。それ以外の場合は、3 値的論理が適用されます。ansinull オプションは、クエリの WHERE 句のこの特別なケースにのみ影響するようになりました。

- **Sybase Central と Interactive SQL が COMMLINKS 接続パラメータを受け入れる** Adaptive Server Anywhere の以前のバージョンでは、Sybase Central と Interactive SQL (dbisql ユーティリティ) は、COMMLINKS 接続パラメータを無視していました。Sybase Central と Interactive SQL は、このパラメータを受け入れるようになりました。

  この変更により、一部の接続文字列の動作が、Adaptive Server Anywhere の以前のバージョンでの動作とは異なる場合があります。具体的には、COMMLINKS=tcpip を指定しないと、Interactive SQL と Sybase Central はネットワーク上のサーバを探しません。

  詳細については、「CommLinks 接続パラメータ [LINKS]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **クライアントは SQLLOCALE 環境変数を無視する** クライアントは CharSet 接続パラメータを使用して接続に使用する文字セットを指定できます。Adaptive Server Anywhere の以前のバージョンでは、CharSet 接続パラメータの指定がない場合、クライアントのデフォルト文字セットの変更には、SQLLOCALE 環境変数の CHARSET パラメータが使用されていました。クライアントはこの SQLLOCALE 環境変数を無視するようになりました。
- **サポートされない文字セットを使用すると接続が失敗する** クライアントは CharSet 接続パラメータを使用して接続に使用する文字セットを指定できます。ただし、サーバが要求された文字セットをサポートしていないと、接続は失敗します。Adaptive Server Anywhere の以前のバージョンでは、クライアントがサポートされていない文字セットを要求すると、警告は表示されましたが、接続は確立されました。クライアントが文字セットを指定せず、サーバがクライアントのローカル文字セットをサポートしていない場合、接続は確立されますが、文字セットがサポートされていないことを知らせる警告が表示されます。

  この動作は、バージョン 6.x、バージョン 7.x、バージョン 8 データベース・サーバに接続するバージョン 8 クライアントで発生します。
- **デフォルト・パケット・サイズの変更** クライアント／サーバ通信のデフォルト・パケット・サイズが、1024 バイトから 1460 バイトに変更されました。

  パケット・サイズの詳細については、「CommBufferSize 接続パラメータ [CBSIZE]」『SQL Anywhere サーバ - データベース管理』と「-p サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **dbdsn ユーティリティが管理するのは Adaptive Server Anywhere データ・ソースのみ** Adaptive Server Anywhere ODBC データ・ソースを管理する dbdsn ユーティリティは、Adaptive Server Anywhere データ・ソースのみを管理するように明示的に制限されるようになりました。
- **login_procedure オプションには DBA 権限が必要** login_procedure オプションは、DBA 権限を持つユーザだけが設定できます。Adaptive Server Anywhere の以前のバージョンでは、このオプションは DBA 権限を持たなくても設定できました。DBA 権限を持つユーザは、他のユーザの設定も変更できますが、DBA 権限を持たないユーザは自分の設定の変更もできません。この変更により、DBA は、必要に応じて、ユーザの接続時に確実に共通のプロシージャを実行することができます。

  詳細については、「login_procedure オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **ESTIMATE_SOURCE が返す新しい値** ESTIMATE_SOURCE 関数は、以前よりも詳細な値を返します。

  詳細については、「ESTIMATE_SOURCE 関数 [その他]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

廃止されサポートされなくなった機能

このリストには、サポートを終了した機能の中で既存のアプリケーションに影響のあるものがまとめられています。

- **NetWare 4.10 のサポート終了**　Novell NetWare バージョン 4.11 以降は引き続きサポートされています。バージョン 3.x と 4.10 はサポートされていません。

- **NetBIOS のサポート終了**　NetBIOS ポートは、サポートされなくなりました。NetBIOS を使用する場合は、TCP/IP または SPX に切り替えてください。

- **IPX のサポート終了**　IPX ポートは、サポートされなくなりました。IPX を使用する場合は、SPX または TCP/IP に切り替えてください。

- **使わなくなった照合**　次の照合は、サポートされなくなりました。別の照合に置き換えられた照合は、その旨明記されています。

使用されなくなった照合	置き換えられた照合
437	437LATIN1
850	850LATIN1
852	852LATIN2
860	860LATIN1
863	863LATIN1
865	865NOR
SJIS	932JPN
SJIS2	932JPN
WIN_LATIN1	1252LATIN1
WIN_LATIN5	1254TRK
Internal	850LATIN1
437EBCDIC	

- **-e オプションのサポート終了**　データ・ソース・ユーティリティでクライアント／サーバ通信を暗号化するために使用されていた -e コマンド・ライン・オプションと -e オプションは、サポートされなくました。代わりに、-ec オプションが使用されます。サーバ側の -ec **simple** は、Adaptive Server Anywhere の以前のバージョンの -e と同じ暗号化アルゴリズムを使用します。

- **None パラメータの廃止**　isql_plan オプションの None パラメータは、サポートされなくなりました。クエリ最適化プランは、[結果] ウィンドウ枠の [プラン] タブに表示されるようになりました。[プラン] タブをクリックすると、常にプランが表示されます。以前は、プランは [メッセージ] ウィンドウ枠に表示されていました。

- **WITH HASH SIZE n 句の廃止** WITH HASH SIZE 句は、サポートされなくなりました。

- **max_work_table_hash_size オプションの廃止** max_work_table_hash_size オプションは、サポートされなくなりました。

- **max_hash_size オプションの廃止** max_hash_size オプションは、サポートされなくなりました。

- **SATMP 環境変数の廃止** Adaptive Server Anywhere の UNIX バージョンで使用されていた、テンポラリ・ファイルの保存ディレクトリを示す SATMP 環境変数は、サポートされなくなりました。UNIX では、ASTMP 環境変数を使用してテンポラリ・ファイルの保存場所を示すことができます。

  詳細については、「SATMP 環境変数」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **dbtran -id オプションの削除** 新しいバージョンでは、dbtran ユーティリティの -id コマンド・ライン・オプションが廃止されました。

Mobile Link の動作の変更

- **Mobile Link Adaptive Server Anywhere クライアントの設定** Mobile Link クライアントは、同期定義ではなく、パブリケーションと同期サブスクリプションを使用して設定されるようになりました。

  詳細については、「SQL Anywhere クライアント」『Mobile Link - クライアント管理』を参照してください。

- **最後ダウンロード・タイムスタンプ・パラメータでスクリプトを変更する** 多数のスクリプトに対応する 1 つの新しいパラメータが追加され、タイムスタンプ・ベースの同期の実装が容易になりました。この新しいパラメータは、多数のスクリプトに対する最初のパラメータとして指定されるため、既存のスクリプトは中断されます。既存スクリプトの使用を続行するには、-zd Mobile Link サーバ・コマンド・ライン・オプションを指定して、最終ダウンロード・タイムスタンプを最後のパラメータとして指定するように動作を変更してください。

- **Mobile Link のシャットダウン** 以前は、*dbmlstop* コマンドを使用して、リモート接続から Mobile Link サーバをシャットダウンできました。現在は、Mobile Link サーバと同じマシンからの *dbmlstop* 要求だけが、Mobile Link サーバをシャットダウンするようになりました。*dbmlstop* にサーバを停止させることを許可する -zs オプションは、必要なくなりました。

- **TCP/IP ストリーム内の活性検出デフォルト設定の変更** keep_alive のデフォルト設定は、1 (ON) になりました。

- **Mobile Link で dbmluser 情報を非表示にできる** dbmluser コマンド・ライン・ユーティリティを使用したときに表示された、タイムスタンプ、著作権、その他の Mobile Link サーバ・メッセージなどの情報は、デフォルトでは表示されなくなりました。

- **Mobile Link ユーザ認証**　Mobile Link ユーザ認証を使用しない場合は、Mobile Link サーバ・コマンドで -zu+ オプションを使用してください。
- **デフォルトのログ拡張子が .mls に変更**　各ファイルには、DDMMYYNN.MLS という名前が付けられるようになりました。DD は日、MM は月、YY は西暦年を表します。NN は、1 から始まるファイルの連続番号です。
- **dbmlsync StreamCompression 拡張オプションの廃止**　このオプションは無視されます。

Ultra Light の動作の変更

- **Palm アプリケーションに必要なコード変更**　Palm Computing Platform バージョン 4.x に標準のレコード・ベースのデータベース記憶領域を使用するか、ファイル・ベースの拡張カード記憶装置を使用するかをコードで指定する必要があります。関数呼び出しを 1 つ追加してから、ULPalmLaunch (Embedded SQL の場合) または ULData.PalmLaunch (C++ API の場合) を呼び出してください。追加する関数呼び出しは次のとおりです。

  ```
  ULEnablePalmRecordDB( &sqlca );
  ```

  または

  ```
  ULEnableFileDB( & sqlca );
  ```

  レコード・ベースの記憶装置を使用する場合は **ULEnablePalmRecordDB** を、ファイル・ベースの記憶装置を使用する場合は **ULEnableFileDB** を指定します。デバイスがファイル・ベースの記憶装置をサポートしていない場合、ULPalmLaunch によって SQLCODE -82 が設定されます。

Ultra Light による以下の環境および機能のサポートは終了しました。

- **DOS ターゲット・プラットフォーム**　DOS は、サポート対象ではなくなりました。
- **Metrowerks CodeWarrior 5 開発プラットフォーム**　Ultra Light を開発するには、CodeWarrior 6 が必要になりました。
- **Palm 2.x のサポート終了**　Ultra Light は、PalmPilot Professional などの Palm OS 2.x デバイスのための開発をサポートしなくなりました。バージョン 3.0 以上が必要です。
- **ULPalmDBStream と ULConduitStream の廃止**　Palm Computing Platform での HotSync または ScoutSync による同期に新しい同期ストリームが導入されたことで、**ULPalmDBStream** 関数と **ULConduitStream** 関数は使用されなくなりました。これらの関数は受け入れられますが、効果はありません。
- **Ultra Light ジェネレータは、外部 Java 仮想マシンを使用する**　Ultra Light アナライザはデータベース・エンジンの外部で実行されるようになったので、Java が有効化されていないリファレンス・データベースに対しても使用できます。

- **Ultra Light JDBC パッケージ名の変更**　Ultra Light JDBC 関数のパッケージ名は、**com.sybase.asa.ultralite.jdbc** から **ianywhere.ultralite.jdbc** に変更されました。したがって、Ultra Light アプリケーションに使用される import 文を変更する必要があります。
- **すべての変更は、ダウンロード同期の前にコミットが必要**　同期させる前にすべての変更をコミットするという規則が、ダウンロード専用同期にも適用されるようになりました。

Adaptive Server Anywhere の動作の変更のいくつかは、アプリケーションに影響を及ぼす可能性もあるので、その内容についても確認してください。

第 9 章

バージョン 7.0.3 の新機能

目次

新機能

この項では、Adaptive Server Anywhere バージョン 7.0.3 の新機能について説明します。新機能の一覧を挙げ、各機能の詳細が説明されている参照先も示します。

- **ブランク埋め込みと大文字と小文字の区別のためのデータベース・プロパティ**　文字列比較時にデータベースでブランク埋め込みを使用するか (BlankPadding)、またはデータベースで大文字と小文字を区別するか (CaseSensitive) を決める 2 つのプロパティが新しく追加されました。

  詳細については、「データベース・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **C2 セキュリティ・モード用のサーバ・プロパティ**　データベース・サーバが -sc オプションを使って起動されたかどうかを判定する、新しい C2 サーバ・プロパティが追加されました。-sc オプションは、C2 基準を満たした環境で使うためのものです。

  詳細については、「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **ログイン・プロシージャで接続のブロックが可能**　login_procedure オプションを使用すると、新しい接続ごとにストアド・プロシージャを呼び出すようにできます。このプロシージャによって、データベース接続を拒否することもできます。

  詳細については、「login_procedure オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **UNIX 上で FileDSN をサポート**　ODBC データ・ソース用の FileDSN 接続パラメータが UNIX でサポートされるようになりました。

動作の変更

ここでは、これまでのバージョンとは異なる動作について説明します。

- **LOAD TABLE のセマンティックの変更**　カラム・リストが指定されたときの LOAD TABLE コマンドのセマンティックが改善されました。カラム・リストには、ファイル内の各カラムを出現順に指定しなければなりません。リストにないカラム名は、カラムの NULL 入力属性やデータ型、デフォルト動作などにより、NULL、ゼロ、空文字列、またはデフォルト値のいずれかに設定されます。

  LOAD TABLE によって無視される入力ファイル内のカラムは、カラム名 **filler()** を使用して指定できます。

  詳細については、「LOAD TABLE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

第 10 章

バージョン 7.0.2 の新機能

目次

バージョン 7.0.2 の新機能

この項では、SQL Anywhere Studio バージョン 7.0.2 のコンポーネントに導入された新機能について説明します。

Adaptive Server Anywhere の新機能

この項では、Adaptive Server Anywhere バージョン 7.0.2 の新機能について説明します。新機能について主要なものからそうでないものまですべてを示し、各機能の詳細についての参照先も記述しています。

- **動的キャッシュ・サイズ決定** Windows 95/98 では、データベース・サーバの負荷とその他のシステム・メモリへの要求に応じて、データベース・サーバのキャッシュ・サイズが増減します。この機能を使用すると、さまざまな状況で明示的にキャッシュ・サイズを選択する必要がなくなり、パフォーマンスが向上します。

  詳細については、「動的キャッシュ・サイズ決定 (Windows)」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **現在のライセンス情報の表示** ライセンス [dblic] ユーティリティでは、サーバを起動せずに、サーバの実行プログラムに関する現在のライセンス情報を表示できる引数を使用できるようになりました。

  詳細については、「サーバ・ライセンス取得ユーティリティ (dblic)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **照合の追加** ロシア語とウクライナ語 (1251CYR、ANSI コード・ページ 1251) をサポートする照合、トルコ語 (1254TRK、ANSI コード・ページ 1254) をサポートする照合、ドイツ語 (1252DEU、ANSI コード・ページ 1252) の一部のユーザをサポートするための特別な照合の 3 つの照合が使用可能になりました。

  ドイツ語では、1252LATIN1 照合を使用することをおすすめします。1252DEU は特別な照合です。この照合は、ソートや比較の特性を理解した上で使用するようにしてください。

  使用可能な照合の完全なリストについては、「照合の選択」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Interactive SQL のリターン・コード** Interactive SQL をコマンド・プロンプトから実行したときに、セッション中の動作の成功または失敗を示すプログラム終了コードを設定できるようになりました。

  詳細については、「Interactive SQL ユーティリティ (dbisql)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **delete_old_logs の強化** delete_old_logs データベース・オプションは、レプリケーション環境でオフライン・トランザクション・ログを管理するときに使用されます。このオプションが、処理済みのトランザクション・ログを削除するときに細かい制御ができるように強化されました。

詳細については、「delete_old_logs オプション [Mobile Link クライアント] [SQL Remote] [Replication Agent]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **接続のトラブルシューティングと強化** クライアント／サーバ通信のトラブルシューティングとチューニングを強化するために、次の変更が行われました。
  - クライアントのデバッグ・ログ・ファイルに、APPINFO 文字列が追加されました。

    詳細については、「AppInfo 接続パラメータ [APP]」『SQL Anywhere サーバ - データベース管理』を参照してください。
  - 新しい 2 つの接続パラメータを使用すると、ローのプリフェッチをチューニングできます。

    詳細については、「PrefetchRows 接続パラメータ [PROWS]」『SQL Anywhere サーバ - データベース管理』と「PrefetchBuffer 接続パラメータ [PBUF]」『SQL Anywhere サーバ - データベース管理』を参照してください。
  - これまで、**ConnectionName** 接続パラメータの値は、ODBC クライアントに対して上書きされていました。今回のリリースでは、ODBC クライアントから **ConnectionName** パラメータを使用できるようになりました。

    接続パラメータのリストについては、「接続パラメータ」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **言語選択ユーティリティ** 言語選択ユーティリティ (dblang) を使用して、Adaptive Server Anywhere のメッセージと Sybase Central インタフェース要素の言語レジストリについてレポートと修正ができます。

  詳細については、「言語選択ユーティリティ (dblang)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **dbspawn の強化** プロセス生成 (dbspawn) ユーティリティでは、データベース・サーバのオペレーティング・システム・プロセス ID をオプションでレポートできます。

  詳細については、「サーバ・バックグラウンド起動ユーティリティ (dbspawn)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **週の最初の曜日オプション** 週の最初の曜日のデフォルトが 7 になりました。これは日曜日に当たります。この値は、平日の値を取得するときの DATEPART の結果に影響します。週の最初の曜日は、Transact-SQL SET 文の DATEFIRST オプションを使用して変更できます。また、SET OPTION first_day_of_week=*n* を使用して、週の最初の曜日を永続的に設定できます。

  詳細については、「SET 文 [T-SQL]」『SQL Anywhere サーバ - SQL リファレンス』または「first_day_of_week オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **新しいマイグレーション・ツール** 新しいストアド・プロシージャのセット sa_migrate を使用して、Oracle、DB2、Microsoft SQL Server、Sybase Adaptive Server Enterprise、Sybase Adaptive Server Anywhere のリモート・データベースを Adaptive Server Anywhere にマイグレート (インポート) できます。

詳細については、「SQL Anywhere へのデータベースの移行」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **イベント・ハンドラ** Adaptive Server Anywhere は、ある時点で実行されている特定のイベント・ハンドラのインスタンス数を決定できるようになりました。この機能を使用して、イベント・ハンドラが一度に 1 つだけ実行されるように制限できます。

  詳細については、「EVENT_PARAMETER 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **新しい接続プロパティ** 新しい接続プロパティによって、イベント・ハンドラを実行するために使用する内部接続を互いに区別できます。

  詳細については、「CONNECTION_PROPERTY 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **dbdsn によるユーザ指定子とシステム指定子のサポート** データ・ソース [dbdsn] ユーティリティは、u (ユーザ) と s (システム) オプションをサポートするようになりました。

  詳細については、「データ・ソース・ユーティリティ (dbdsn)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **@filename ファイル内のコメントのサポート** Adaptive Server Anywhere は、@filename ファイルにあるコメント行をサポートするようになりました。

  詳細については、「@data サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **タイムスタンプ・トランケート・オプション** Adaptive Server Anywhere 以外のデータベースとの互換性を高めるために、タイムスタンプ値をトランケートできるようになりました。

  詳細については、「truncate_timestamp_values オプション [データベース] [Mobile Link クライアント]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **ライセンス情報の取得** 使用中の Adaptive Server Anywhere に関する正確なライセンス情報を取得できるように、エンジン・プロパティが追加されました。

  詳細については、「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **オートインクリメント値のリセット** sa_reset_identity システム・プロシージャを使用すると、次のローのオートインクリメント値をリセットできます。

  詳細については、「sa_reset_identity システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

Mobile Link の新機能

- **アップロード・ストリームを適用するスレッドの最大数** データベース競合を極力避けるために、-wu コマンド・ライン・オプションを使用して、同時アップロードできるワーカ・スレッ

ドの最大数を設定できるようになりました。アップロード要求は、要求を受け付けた順序で処理されます。

- 詳細については、「Mobile Link サーバのオプション」『Mobile Link - サーバ管理』を参照してください。

バージョン 7.0.2 での動作の変更

この項では、SQL Anywhere Studio バージョン 7.0.2 のコンポーネントに導入された動作の変更について説明します。

Adaptive Server Anywhere の動作の変更

ここでは、これまでのバージョンとは異なる動作について説明します。

- **エイリアスは最初の参照よりも前に定義する必要あり**　SQL Anywhere の以前のバージョンでは、エイリアスが定義される前に、そのエイリアスを SELECT リストで参照できました。今回のリリースでは、そのような操作を行うと、エラー「エイリアス alias 名の定義は、最初の参照前に記述する必要があります。」が発生します。このエラーを避けるために、SELECT リストを並べ替えて、エイリアスが定義されてから最初の使用が行われるようにする必要があります。

第 11 章

バージョン 7.0.1 の新機能

目次

バージョン 7.0.1 の新機能

この項では、SQL Anywhere Studio バージョン 7.0.1 のコンポーネントに導入された新機能について説明します。

Adaptive Server Anywhere の新機能

この項では、Adaptive Server Anywhere バージョン 7.0.1 の新機能について説明します。新機能について主要なものからそうでないものまですべてを示し、各機能の詳細についての参照先も記述しています。

- **新しいサービス ユーティリティ**　データベース・サーバを NT 環境下でサービスとして実行させることで、データベースはそれが実行されているコンピュータに制限されることなく実行を続けることができます。これまでは、サービスの追加は、Sybase Central から [新しいサービスの作成] ウィザードを使用して行っていました。Adaptive Server Anywhere のバージョン 7 では、サービス作成 [dbsvc] ユーティリティを使用して、Windows NT 上で Adaptive Server Anywhere サービスを管理することもできるようになりました。さまざまなオプションを使用して、サービスの追加と削除、すべての Adaptive Server Anywhere サービスのリスト、または特定のサービスの詳細表示ができます。この機能は、インストール環境でサービスの作成を埋め込むときに特に便利です。

  サービス・ユーティリティの詳細については、「Windows 用サービス・ユーティリティ (dbsvc)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Windows CE 3.0 のサポート**　Adaptive Server Anywhere は、Windows CE 2.11 のほかに、以下のプロセッサ上で動作する Windows CE 3.0 もサポートするようになりました。

  - MIPS
  - Hitachi SH3
  - ARM

  Windows CE 2.11 は広範囲のプラットフォームでサポートされます。

  Windows CE 3.0 のサポートによって、CE 上の OLE DB ドライバは、追加ソフトウェアをインストールしなくても動作します。

- **Embedded SQL の強化**　新しい関数 **db_locate_servers** は、TCP/IP 上で受信している Adaptive Server Anywhere データベース・サーバの場所をプログラムによって検出します。

  詳細については、「db_locate_servers 関数」『SQL Anywhere サーバ - プログラミング』を参照してください。

  新しいコールバック関数 DB_CALLBACK_CONN_DROPPED を使用すると、データベース・サーバの接続が切断されるときのロジックを追加できます。

詳細については、「db_register_a_callback 関数」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **接続レベルのデバッグと LogFile 接続パラメータ** クライアント側の接続パラメータである DBG と LOG は接続に固有となったので、1 つのアプリケーションからでも、接続ごとにデバッグ情報を設定できます。

  詳細については、「LogFile 接続パラメータ [LOG]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しいデータベース・プロパティ** Replication Agent または LTM のユーザのために、**LTMGeneration** プロパティが追加されました。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。

  詳細については、「データベース・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しい配備機能** InstallShield Professional 5.5 以降を使用している場合は、新しい SQL Anywhere Studio InstallShield Template Projects を使用して、作成したアプリケーションを配備できます。これによって、テンプレート・プロジェクトの全部またはインストールに適用する部分だけを使用して、アプリケーションのインストール・プログラムを迅速に構築できます。

- **Backup 文の新機能** Backup 文を使用するときに、ディレクトリに空の文字列を指定することで、ログをコピーしないで名前を変更したり、トランケートしたりできます。これは、スペースが問題となるレプリケーション環境では特に便利です。この機能をトランザクション・ログ・サイズのイベント・ハンドラと一緒に使用して、ログが所定のサイズに達したときに名前を変更したり、delete_old_logs オプションと一緒に使用して、ログが必要なくなったときに削除したりすることができます。

  詳細については、「BACKUP 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

Mobile Link の新機能

次に、バージョン 7.0.1 で導入したソフトウェアに加えられた変更と追加を示します。

- **ユーザの認証** パスワードに基づくユーザ認証システムによって、Mobile Link のインストール環境のセキュリティがさらに強化されます。

  詳細については、「Mobile Link ユーザ」『Mobile Link - クライアント管理』を参照してください。

- **トランスポート・レイヤ・セキュリティ・マニュアルの増補** トランスポート・レイヤ・セキュリティのマニュアルが改訂され、この強力なセキュリティ・メカニズムで構築可能なさまざまなアーキテクチャの説明が加わりました。

  詳細については、「トランスポート・レイヤ・セキュリティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **同期と同期関連プロセスのカスタマイズ**　Adaptive Server Anywhere 同期クライアント *dbmlsync* が、一連のイベントをサポートするようになりました。Adaptive Server Anywhere データベースにストアド・プロシージャを追加して、イベント・ベースのアクションをプログラムできます。これによって、同期処理に柔軟性が加わります。同期のスケジュール機能もその 1 つです。

  詳細については、「dbmlsync のフックの概要」『Mobile Link - クライアント管理』を参照してください。

- **同期の最適化**　同期処理の以下の点を最適化できます。
  - Ultra Light クライアント・アプリケーションでは、同期にアップロードだけを含み、ダウンロード・フェーズを行わないように指定できます。

    アップロードだけが必要な場合は、このオプションによって、同期にかかる時間が短縮されます。
  - Adaptive Server Anywhere クライアントでは、インクリメンタル・アップロード・オプションを指定して、多くのアップロードを行う際に必要となるメモリを少なくできます。

    詳細については、「dbmlsync 拡張オプションの使用」『Mobile Link - クライアント管理』を参照してください。
  - Adaptive Server Anywhere クライアントでは、同期中にローの同時修正が可能です。

    詳細については、「同期中の同時実行性」『Mobile Link - クライアント管理』を参照してください。
- **同期のスケジュール**　*dbmlsync* ユーティリティの拡張オプションまたは同期定義を使用すると、スケジュールに基づいて同期が実行できます。

  詳細については、「同期のスケジュール」『Mobile Link - クライアント管理』を参照してください。
- **Adaptive Server Anywhere クライアントの同期ユーティリティの強化**　*dbmlsync* ユーティリティの機能が、一部強化されました。
  - -mp と -mn オプションを使用すると、Mobile Link パスワードの指定または変更ができます。
  - -n オプションを繰り返し指定すると、複数の同期定義を同期できます。
  - -v オプションで、より便利な情報を得ることができるようになりました。同期定義内に設定されているオプションがその例です。
  - -r オプションが拡張され、クライアントと統合データベース内に記録された進行状況インジケータが一致しない場合のアップロードをさらに柔軟にできるようになりました。
  - -x オプションで、トランザクション・ログの名前を変更し、再起動できます。このオプションは、クライアント側のバックアップが不要になるように、クライアントのデータのバックアップとして統合データベースを使用する場合に役に立ちます。

- コマンド・ラインで接続パラメータを指定しない場合、接続パラメータと起動オプションを指定できるダイアログを *dbmlsync* が表示します。
- *dbmlsync* ウィンドウでは、同期の進行状況が表示され、同期をキャンセルできます。

  詳細については、「dbmlsync 構文」『Mobile Link - クライアント管理』を参照してください。

- **Mobile Link サーバの新しいオプション**　Mobile Link サーバでは、オプションが追加されました。

  詳細については、「Mobile Link サーバのオプション」『Mobile Link - サーバ管理』を参照してください。

- **新しいスクリプト・イベント**　ODBC Driver Manager で発生するエラーを処理し報告するための新しいスクリプトが追加されました。これによって、同期テクニックを設計するときの柔軟性が増加します。

  詳細については、次の項を参照してください。

  - 「handle_odbc_error 接続イベント」『Mobile Link - サーバ管理』
  - 「prepare_for_download 接続イベント」『Mobile Link - サーバ管理』
  - 「report_odbc_error 接続イベント」『Mobile Link - サーバ管理』

- **dbmlsync 機能へのインタフェース**　プログラミング言語 C を使用している開発者は、*dbmlsync* ユーティリティの機能をアプリケーションに追加できます。

  詳細については、「アプリケーションからの同期の開始」『Mobile Link - クライアント管理』を参照してください。

SQL Remote の新機能

SQL Remote バージョン 7.0.1 に、次の新機能が追加されています。

- **Novell NetWare 上のメッセージ・リンクの追加**　Novell NetWare で FTP と SMTP/POP メッセージ・リンクを使用できるようになりました。
- **冗長モードの強化**　Message Agent の冗長モードで、ユーザ ID とアスタリスクに置換されたパスワードを含むすべての接続情報が出力されるようになりました。

Ultra Light の新機能

Ultra Light 7.0.1 には、次の新機能が導入されています。

- **Palm Computing Platform 用の新しい同期ストリーム**　今回のリリースでは、現行の **ULPalmDBStream** 同期ストリームに加え、新しい同期ストリームを Palm Computing Platform

に使用できます。この新しいストリームは **ULConduitStream** と呼ばれ、ほとんどの状況で、HotSync 同期のパフォーマンスを大幅に向上させることができます。

この機能は使用されなくなりました
バージョン 8.0.0 から導入された新しい conduit ベースの同期ストリームが、**ULPalmDBStream** と **ULConduitStream** の両方に取って代わります。

- **同期のモニタとキャンセル** 同期状況を表示し、同期をキャンセルできる機能を Ultra Light アプリケーションに構築できます。
- **Mobile Link でのユーザ認証** Mobile Link 同期独自のユーザ認証スキームが使用されるようになりました。このスキームを利用するために、Ultra Light 同期パラメータにパスワード用のフィールドとメソッドが追加されました。

  詳細については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」『Mobile Link - クライアント管理』を参照してください。
- **セキュリティ機能を備えた同期のための新しいプラットフォーム** Hitachi SH4 チップを搭載した Windows CE、Intel x86 チップまたは Windows VxSim エミュレータを搭載した VxWorks などの広範囲のターゲット・プラットフォームから、トランスポート・レイヤのセキュリティ機能を同期に使用できるようになりました。

  詳細については、次の項を参照してください。

  - 「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」『Mobile Link - クライアント管理』.
  - 「Windows CE での同期」『Ultra Light - C/C++ プログラミング』.

  バージョン 9 における VxWorks の非サポート
  バージョン 9 では VxWorks プラットフォームはいっさいサポートされません。
- **非同期テーブル** Ultra Light データベースに含めるものの同期しないテーブルをリファレンス・データベースの中に入れることができます。これらのテーブルは、同期以外はリモート・データベース内の他のテーブルと同じように使用できます。
- **Windows CE エミュレータのサポートの強化** Ultra Light アプリケーションを Windows CE x86 エミュレータで実行できるようになりました。
- **同期の最適化** クライアント・アプリケーションでは、同期にアップロードだけを含み、ダウンロード・フェーズを行わないように指定できます。アップロードだけが必要な場合、特に低速な通信リンクを使用する場合は、このオプションによって同期にかかる時間を短縮できます。

  詳細については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」『Mobile Link - クライアント管理』を参照してください。

- **HTTP バージョンの自動検出**　Mobile Link サーバは、各クライアントが使用している HTTP バージョンを検出してそれを使用するようになりました。この機能により、Mobile Link サーバの -x オプションで使用されていた **version** パラメータが不要になります。

  Mobile Link サーバ・コマンド・ラインの詳細については、「Mobile Link サーバのオプション」『Mobile Link - サーバ管理』を参照してください。

- **クライアント・ポートの指定**　同期時にクライアントが使用するポートの範囲を、クライアント側で指定できます。この機能は、ファイアウォールの内側にあるクライアントからファイアウォールの外側にある Mobile Link サーバに対して同期するときに便利です。

  詳細については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」『Mobile Link - クライアント管理』を参照してください。

第 12 章

バージョン 7.0.0 の新機能

目次

バージョン 7.0.0 の新機能

マニュアルの基本フォーマットは HTML ヘルプです。HTML ヘルプのホームページからは、新機能、Sybase への問い合わせ方法、その他、本リリースの導入時に必要な情報に簡単にアクセスできます。

使用するコンピュータに Internet Explorer 4.0 も HTML ヘルプもインストールされていない場合は、HTML ヘルプではなく Windows ヘルプをインストールして使用してください。Windows ヘルプには HTML ヘルプのホームページは表示されませんが、その他の内容は同じです。

Windows ヘルプを使用する場合は、『*Adaptive Server Anywhere 入門*』の第 1 章に Adaptive Server Anywhere の新機能が記載されています。また、『*Ultra Light 開発者用ガイド*』と『*レプリケーションおよび同期ガイド*』の第 1 章に、それぞれのテクノロジの新機能に関する情報が記載されています。

Adaptive Server Anywhere の新機能

この項では、Adaptive Server Anywhere バージョン 7.0 の新機能について説明します。新機能について主要なものからそうでないものまですべてを示し、各機能の詳細についてのマニュアル内の参照先も記述しています。

このマニュアルの製本版を使用していて、SQL Anywhere Studio のマニュアル・セット全巻をお持ちでない場合、各機能の詳細についてはオンライン・マニュアルを参照してください。オンライン・マニュアルで情報を検索するには、目次を開き、参照したい項目を入力します。

管理機能と操作性の強化

- **データベースのタスク・スケジューリングとイベント処理**　データベースに操作のスケジュールを追加できるようになりました。この機能は、自動バックアップ、サマリ・テーブルに入力する定期レポートなどのタスクを実行するのに便利です。

  データベース・サーバに対して、特定のイベントが発生したときにイベント・ハンドラを起動するように指示できます。イベントにはデータベース・ファイルまたはトランザクション・ログ・ファイルを格納するドライブのディスク領域の閾値超過、接続試行の失敗などが含まれます。

  イベント・ハンドラの作成と変更には Sybase Central を使用でき、デバッグには Adaptive Server Anywhere デバッガを使用できます。

  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」『SQL Anywhere サーバ - データベース管理』と「CREATE EVENT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **更新された Sybase Central**　Sybase Central が書き直され、重要な新機能が追加されました。特に重要なのは、Windows オペレーティング・システムだけではなく、サポートされているすべてのプラットフォームから Sybase Central を使用できるようになったことです。

- **更新された Interactive SQL** Interactive SQL [dbisql] ユーティリティが強化され、サポートされているすべてのプラットフォームからウィンドウ・アプリケーションとして使用できるようになりました。
- **新しい検証機能** データベースの追加検証機能として、VALIDATE INDEX 文が新しく加わり、VALIDATE TABLE 文が強化されました。この文は、検証 [dbvalid] ユーティリティと sa_validate システム・プロシージャの両方で呼び出すことができ、どちらの場合でも新しい検証機能を使用できます。

  詳細については、「VALIDATE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **ロックのトラブルシューティング** 新しいシステム・プロシージャ sa_locks を使うと、データベースのロック情報を通知できます。ロック状態が検出されると、関連する接続プロセス情報を **AppInfo** 接続プロパティを使って表示できます。

  詳細については、「sa_locks システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』と「AppInfo 接続パラメータ [APP]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **結果セットのアンロード** 新しい UNLOAD SQL 文を使用すると、クエリ結果セットをカンマ区切りのテキスト・ファイルにアンロードできます。

  詳細については、「UNLOAD 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **データベースのバックアップ・コピーの検証** WAIT BEFORE START 句を使ってデータベースのバックアップを作成すると、読み取り専用モードで開き、検証ができるバックアップ・コピーが作成されます。

  詳細については、「BACKUP 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **デフォルトのグローバル・オートインクリメント** この機能を使用すると、SQL Remote レプリケーション環境にあるすべてのデータベースにユニークな整数キーを簡単に生成できます。

分散コンピューティング・アーキテクチャとの統合

- **分散トランザクションと 3 層コンピューティング** 分散トランザクションには、単一のトランザクションで複数のサーバを使用する操作が含まれます。トランザクション・サーバは、分散トランザクションのコミットとロールバック動作を制御します。

  Adaptive Server Anywhere のこのリリースは、DTC (Microsoft Distributed Transaction Coordinator) で管理される分散トランザクションにサーバとして追加できます。Sybase Enterprise Application Server や Microsoft Transaction Server などの製品は、DTC をトランザクションの調整に使用できるので、DTC をサポートすることにより、Adaptive Server Anywhere をこれらの製品を使った 3 層コンピューティングに使用できます。

  詳細については、「3 層コンピューティングと分散トランザクション」『SQL Anywhere サーバ - プログラミング』を参照してください。

COM の統合

- **OLE DB プロバイダ**　OLE DB は Microsoft が提供するデータ・アクセス・モデルです。OLE DB は、Component Object Model (COM) インタフェースを使用します。ODBC と違って、OLE DB は、データ・ソースが SQL クエリ・プロセッサを使用することを仮定していません。これまでも Microsoft が提供する OLE DB/ODBC ブリッジを使用し、OLE DB 経由で Adaptive Server Anywhere にアクセスすることは可能でしたが、今回のリリースから Adaptive Server Anywhere に OLE DB プロバイダが付属するようになりました。このプロバイダには、以下のような利点があります。

  - OLE DB は、今後予定されている Windows CE 版への主要データ・アクセス・ポイントとなります。
  - カーソルによる更新など、OLE DB/ODBC ブリッジを使用している場合には利用できない機能がいくつかあります。
  - Adaptive Server Anywhere の OLE DB プロバイダを使用する場合、配備に ODBC は必要ありません。

  詳細については、「SQL Anywhere OLE DB と ADO API」『SQL Anywhere サーバ - プログラミング』を参照してください。

コネクティビティの強化

- **Java コネクティビティの機能の向上**　jConnect を使用して Java アプリケーションから Adaptive Server Anywhere に接続する場合、かつては ODBC と Embedded SQL アプリケーションでのみ使用可能だったデータベース・サーバの自動スタート、プロトコル・オプションを使ったネットワーク通信の詳細な制御などのさまざまな機能を使用できるようになりました。

- **TCP/IP コネクティビティ**　TCP/IP を使ったクライアント／サーバ接続の確立が簡単になりました。クライアントは、サーバがデフォルトのポート番号 (2638) 以外のポートで実行していても、接続を試行する際にポート番号を指定する必要がなくなりました。データベース・サーバを起動したとき、デフォルトのポート番号がすでに使用されていたら、サーバはオペレーティング・システムから未使用のポートを取得します。

  ファイアウォールを通過する接続を試み (UseUDP=NO を使用)、データベース・サーバがポート 2638 を使用していない場合は、今までどおりポート番号を指定する必要があります。この方法の詳細については、「ファイアウォール経由の接続」『SQL Anywhere サーバ - データベース管理』を参照してください。

  サーバ列挙ユーティリティ (dblocate) ユーティリティによって、ネットワーク上で TCP/IP を実行するすべての Adaptive Server Anywhere データベース・サーバが表示されます。詳細については、「サーバ列挙ユーティリティ (dblocate)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **SPX コネクティビティ**　データベースへの接続に SPX プロトコルを使用できます。この機能は、IPX/SPX をプライマリ・ネットワーク・プロトコルとして使用する Novell NetWare 環境では特に役に立ちます。IPX よりも SPX の使用をおすすめします。

  クライアント側の SPX の詳細については、「CommLinks 接続パラメータ [LINKS]」『SQL Anywhere サーバ - データベース管理』を参照してください。サーバ側の SPX の詳細につい

ては、「-x サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。SPX で使用するネットワーク・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」『SQL Anywhere サーバ - データベース管理』を参照してください。

パフォーマンスの強化

- **動的キャッシュ・サイズ決定** Windows NT と UNIX では、データベース・サーバの負荷とその他のシステム・メモリへの要求に応じて、データベース・サーバのキャッシュ・サイズが増減します。この機能を使用すると、さまざまな状況で明示的にキャッシュ・サイズを選択する必要がなくなり、パフォーマンスが向上します。Windows 95/98 にもキャッシュのサイズ変更機能が実装されていますが、基本機能に限られます。

  詳細については、「パフォーマンス向上へのキャッシュの使用」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **インデックス処理の強化** インデックスに格納される情報量である「**ハッシュ・サイズ**」を、より柔軟に制御できるようになり、インデックスの選択性が高くなりました。また、プライマリ・キー・インデックスと外部キー・インデックスのアーキテクチャが変更になりました。

  複数カラムのインデックス付け、または多くのローで最初の文字セットまたは数字セットが同じであるカラムのインデックス付けについては、ハッシュ・サイズを制御することで、インデックスの選択性が高くなり、パフォーマンスが向上します。

  詳細については、「インデックスの使用」『SQL Anywhere サーバ - SQL の使用法』、「CREATE INDEX 文」『SQL Anywhere サーバ - SQL リファレンス』、「CREATE TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  インデックス・レベル数を確認する方法については、「sa_index_levels システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  これまでのリリースでは、プライマリ・キーと外部キーには、すべてのプライマリ・キー値とそれに関連するすべての外部キー値を示す単一のインデックスが自動的に関連付けられました。このアーキテクチャは、状況によってはパフォーマンスが低下することがありました。新しいインデックス編成では、これらのインデックスを分離するので、状況によってパフォーマンスが向上します。

  キー・インデックスの詳細については、「キーを使ったクエリのパフォーマンス改善」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

  可変ハッシュ・サイズのインデックスと個別キー・インデックス機能を活用するには、データベースをアンロードし、再ロードする必要があります。アップグレード [dbupgrad] ユーティリティの実行では不十分です。

- **文字列拡張用の個別記憶領域** 255 文字を超える値の物理記憶領域が再編成されました。1 つのテーブルに対して割り付けられるページが、2 つのセットに分割されるようになりました。最初のセットには、ローだけが格納されます。ローのカラム値に 255 文字を超える文字列が含まれている場合は、文字列のプレフィクス (255 文字まで) と文字列拡張に対する参照だけがローに格納されます。255 文字を超える文字列では、文字列拡張はテーブル・ページ

の 2 番目のセットに割り付けられます。この変更により、長い値を格納するテーブル・スキャンを要求するクエリのパフォーマンスが向上します。これは、テーブルの逐次スキャンで、最初のセットに含まれているページだけをスキャンすれば済むからです。

この機能を使用するには、データベースを一度アンロードして、再ロードする必要があります。

- **新しいデータベースのページ・サイズ** 1 K、2 K、4 K のページ・サイズに加えて、8 K、16 K、または 32 K のページ・サイズを持つデータベースを作成できるようになりました。

  ページ・サイズを大きくすると、特にデータベースのサイズが大きい場合など、状況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、ページ・サイズを大きくすると、追加メモリが必要になるので、対費用効果を考えて実行してください。

  詳細については、「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバ - データベース管理』と「CREATE DBSPACE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  テーブルごとのインデックス数と、それがページ・サイズによってどう変わるかについては、「SQL Anywhere のサイズと数の制限」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **オプティマイザのチューニング** optimization_goal オプションを使うと、オプティマイザに対して、クエリの最初のローを返すまでの時間、またはすべてのローを返すのに要する総時間を最適化するように指示できます。デフォルトでは、最初のローを最適化します。PowerBuilder DataWindow などの完全な結果セットを要求するアプリケーションを使用する場合は、このオプションの設定を変更してください。

  詳細については、「optimization_goal オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **オプティマイザの強化** オプティマイザがさらに強化され、内部テンポラリ・テーブルを使用し、プライマリ・キーと外部キーのインデックスを使用するクエリのパフォーマンスを最適化できるようになりました。これらの強化された機能を使用するのにユーザによるアクションは必要ありません。

その他の機能強化

- **多数のユーザとその他の識別子** システム・テーブルでデータベース・オブジェクトを区別する多くの識別子が、SMALLINT から UNSIGNED INTEGER に変更されました。この変更により、絶対制限値に違反しないでデータベースに格納できるオブジェクトの数が増えました。

- **イメージとドキュメントの挿入とエクスポート** 2 つの新しいシステム外部関数を使って、ファイルの内容の読み込みと書き込みができます。これらの関数は、Interactive SQL などの環境からテーブルに、イメージやドキュメントなどを直接挿入できます。

  詳細については、「ドキュメントとイメージの挿入」『SQL Anywhere サーバ - SQL の使用法』、「xp_read_file システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』、

「xp_write_file システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **外部関数用の新しいインタフェース** 外部ライブラリを参照するストアド・プロシージャとユーザ定義の関数用のインタフェースが新しくなりました。新しいインタフェースでは、さまざまなオペレーティング・システム (UNIX を含む) とさまざまなデータ型を使用でき、255 バイトまでのデータを返すという制限が取り除かれ、NULL が引数の有効な値として認識されます。これまでのインタフェースもサポートしていますが、新規の開発環境では使用しないでください。

  詳細については、「外部呼び出しを使ったプロシージャと関数の作成」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **START DATABASE、STOP DATABASE、STOP ENGINE 文** これらの文は、これまで Interactive SQL からのみ使用可能でした。現在はどのアプリケーションからでも使用できます。

  詳細については、「START DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』、「STOP DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』、「STOP ENGINE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **更新と削除における FIRST 句と TOP 句** FIRST 句と TOP 句は、WHERE 句の条件を満たす最初の 1 つ以上のロー・セットのみを更新または削除するために使用できます。

  詳細については、「DELETE 文」『SQL Anywhere サーバ - SQL リファレンス』と「UPDATE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **テーブルの明示的ロック** LOCK TABLE 文を使用すると、現在の独立性レベルに関わりなく、テーブル・レベルの同時実行性を直接制御できます。

  詳細については、「LOCK TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **Transact-SQL 外部ジョイン式** WHERE 句のジョイン演算子 (*= と =*) を使用すると、Transact-SQL 構文を使用したいユーザが外部ジョインを指定できるようになります。これまでのリリースでは、そのようなジョインで使用できるのはカラム名だけでした。現在では、ジョイン演算子の双方が同じテーブルを参照するかぎり、どの式でも使用できます。たとえば、次のクエリも現在は使用可能です。

  ```
  select *
  from customer, sales_order
  where substr( customer.id, 1, 1 ) *=
        substr( sales_order.cust_id, 1, 1)
  ```

- **変数を参照できるストアド・プロシージャのカーソル** ストアド・プロシージャとユーザ定義の関数では、次の構文を使って変数にカーソルを宣言できます。

  DECLARE *cursor-name* **CURSOR USING** *variable-name*

  ここで、*variable-name* は、カーソルの SELECT 文を含む文字列変数です。

詳細については、「DECLARE CURSOR 文 [ESQL] [SP]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **データベース・サーバの追加プロパティ** 次のプロパティが追加されました。
  - **PageSize** データベース・サーバは、起動時から終了するまで、同じページ・サイズを使用します。このページ・サイズは、データベース・サーバによってマウントできる最大のページ・サイズです。このページ・サイズは、PageSize サーバ・レベル・プロパティ関数を使用してチェックできるようになりました。

    ```
    select property( 'PageSize' )
    ```

  - **AppInfo** この関数を使用すると、クライアント・アプリケーションの識別情報を取得できます。これは接続プロパティです。

    ```
    select connection_property( 'AppInfo' )
    ```

    詳細については、「AppInfo 接続パラメータ [APP]」『SQL Anywhere サーバ - データベース管理』を参照してください。

  - **IsRuntimeServer** この関数は、データベース・サーバが機能が制限されたデスクトップ・ランタイム・パーソナル・データベース・サーバである場合は、YES を返します。それ以外の場合は、NO を返します。

  - **ログ・トランケーション・ポイント** レプリケーション固有のログ・オフセットに対するプロパティが追加されました。プロパティ **LTMTrunc**、**RemoteTrunc**、**SyncTrunc** は、それぞれ Replication Agent、SQL Remote、Mobile Link *dbmlsync* レプリケーション用の確認された最小のログ・オフセットを返します。これらのオフセットは、トランザクション・ログをトランケートする場所を示すので、トランケーション・ポイントとも呼ばれます。プロパティ **CurrentRedoPos** は、ログ・ファイルの現在のオフセットを返します。そこから次のデータベース操作のログが作成されます。

    すべてのプロパティ関数のリストとそれらにアクセスする方法については、「データベース・プロパティの概要」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **コミット前の参照整合性チェック** 新しいシステム・プロシージャ (sa_check_commit) を使用すると、データベースへの変更をコミットする前に参照整合性の不一致をチェックできます。

  詳細については、「sa_check_commit システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **SQL 関数の強化** 以下の関数が追加または強化されました。
  - **REPLACE 関数** この新しい関数は、すべての部分文字列を別の部分文字列に置き換えます。

    詳細については、「REPLACE 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

  - **LIST 関数の強化** LIST 関数がオプションで 2 番目の値を受け入れられるようになりました。これは、リスト項目を区切るデリミタ文字列です。

詳細については、「LIST 関数 [集合]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **出力リダイレクションの変更** Interactive SQL の出力リダイレクション機能が拡張され、3 つの新しい Interactive SQL 文と、[ファイル] メニューに [エクスポート] オプションが加わりました。

  OUTPUT TO 文を使って、[結果] ウィンドウ枠の内容を新しいファイルにリダイレクトできるようになりました。APPEND 句を加えると、内容が既存のファイルの最後に追加され、また VERBOSE 句を加えると、出力に [メッセージ] ウィンドウ枠の内容が入ります。

  これまでのバージョンでは、Interactive SQL で出力をリダイレクトできるのは、記号 >#、>>#、>&、>>& を使用した場合にかぎられました。これらの記号は現在でも使用できますが、新しい Interactive SQL 文を使った方が、出力をより詳細に指定でき、読みやすいコードが提供されています。

  詳細については、「クエリ結果のエクスポート」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **Embedded SQL の強化** 指定した現在の接続文字列でデータベース・サーバにアクセスできるかどうかをテストするための新しい関数 **db_string_ping_server** が導入されました。

  詳細については、「db_string_ping_server 関数」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **新しい LOAD TABLE/UNLOAD TABLE フォーマット** UNLOAD TABLE 文にデータを BCP フォーマットで出力できる新しいフォーマットが追加され、LOAD TABLE 文に Adaptive Server Enterprise が生成した BLOB を含む BCP 出力ファイルをインポートするための新しいフォーマットが追加されました。

  詳細については、「LOAD TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』または「UNLOAD TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **デフォルトの最終タイムスタンプ** 新しいグローバル変数 **@@dbts** は、DEFAULT TIMESTAMP を使ってカラム用に生成された最終値を表す TIMESTAMP 値を返します。

  詳細については、「グローバル変数」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **トラブルシューティングの強化** データベース・サーバの起動時、-zr オプションを使用すると、サーバが実行する操作のログをファイルに記録できます。sa_server_option プロシージャを使用すると、サーバの実行中に同じ動作を制御できます。

  詳細については、「sa_server_option システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』と「-zr サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **NetWare でのアーカイブのバックアップ** アーカイブのバックアップ・フォーマットが、NetWare でもサポートされるようになりました。テープへのアーカイブのバックアップには NetWare 5 が必要です。

  詳細については、「BACKUP 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **dbtran の追加フィルタ** コマンド・バージョンのログ変換 [dbtran] ユーティリティを使用して、出力をさらにフィルタできるようになりました。

  詳細については、「ログ変換ユーティリティ (dbtran)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **テーブル・トランケーションの高速化** バージョン 7.0 データベースの外部キー付きテーブルに対する TRUNCATE TABLE 文の実行がより高速になりました。

- **イベント・ログ・メッセージを非表示にする** データベース・サーバを Windows NT サービスとして実行する場合、レジストリ・エントリを変更すれば、イベント・ログ・メッセージを非表示にできます。

  詳細については、「Windows イベント・ログ・メッセージを出力しない」『SQL Anywhere サーバ - データベース管理』を参照してください。

SQL Remote の新機能

SQL Remote バージョン 7.0 には、次の新機能が追加されています。

- **グローバルに固有のプライマリ・キーを使用する** Adaptive Server Anywhere データベースの DEFAULT GLOBAL AUTOINCREMENT カラムを各データベースの global_database_id オプション設定とともに使用すると、複数の Adaptive Server Anywhere データベースを対象とする SQL Remote 全体に対して固有のプライマリ・キーを使用できます。この方法は、一部にユーザ設定が必要となるプライマリ・キー・プールよりも便利です。

  詳細については、「グローバル・オートインクリメント・デフォルト・カラム値の使用」『SQL Remote』を参照してください。

- **dbxtract の内部アンロード** 抽出 [dbxtract] ユーティリティは、処理が遅い OUTPUT 文の代わりに、Adaptive Server Anywhere バージョン 7.0 にデフォルトとして組み込まれた UNLOAD 文を使用するようになりました。内部 (サーバ側) と外部 (クライアント側) のアンロードとロードの組み合わせを選択するためのオプションが組み込まれました。

  オプションの完全なリストについては、「抽出ユーティリティ」『SQL Remote』を参照してください。

Mobile Link と Ultra Light の新機能

バージョン 6.0.3 以降のソフトウェアへの変更点と追加点を次に示します。

- **Adaptive Server Anywhere クライアント**　Mobile Link テクノロジが、Ultra Light アプリケーションに加え、Adaptive Server Anywhere もクライアントとしてサポートするようになりました。

  詳細については、「SQL Anywhere クライアント」『Mobile Link - クライアント管理』を参照してください。

- **Adaptive Server Anywhere クライアント・データベースを作成する mlxtract**　*mlxtract* は、Adaptive Server Anywhere データベースを、Mobile Link クライアントとしての使用に適するように作成します。このとき、Adaptive Server Anywhere リファレンス・データベースをテンプレートとして使用します。

- **同期スクリプト・バージョン**　同期スクリプトは、各スクリプトにスクリプト・バージョン名を割り当てればグループ化できるようになりました。この機能を使用すると、Mobile Link サーバは、何種類ものアプリケーションを同期しているときや、同じアプリケーションでもバージョンが違うものを同期しているときも、それぞれ別々に応答できます。

- **新しいデータ型**　Mobile Link テクノロジを使用して LONG BINARY と LONG VARCHAR データ型をレプリケートできるようになりました。

- **新しい HotSync conduit**　新しい HotSync conduit を使用すると、中央の Mobile Link サーバと HotSync を同期できます。Mobile Link サーバが HotSync マネージャと同じコンピュータにある必要がなくなりました。

- **ScoutSync conduit**　Palm Computing Platform 用の Ultra Light アプリケーションが、Riverbed Technologies の ScoutSync テクノロジを使用して同期できるようになりました。

- **report_error スクリプト**　この新しいスクリプトは、同期中のエラーをレポートするのに便利です。また、**report_error** スクリプトを使用すると、**handle_error** スクリプトの動作もより簡単にデバッグできます。**report_error** スクリプトのパラメータは、**handle_error** スクリプトと同じですが、例外として、最初のパラメータは、**handle_error** が返したアクション・コードになります。

バージョン 7.0.0 での動作の変更

この項では、使わなくなった機能、サポートしなくなった機能、これまでのバージョンとは異なる動作について説明します。

Adaptive Server Anywhere の動作の変更

廃止されサポートされなくなった機能

このリストには、サポートを終了した機能の中で既存のアプリケーションに影響のあるものがまとめられています。

- **Windows 3.x と Windows CE 2.0 のサポート終了** Windows 3.1 と Windows 3.11 はサポートされなくなりました。Windows CE 2.0 はサポートされなくなりました。
- **DDE プロトコルのサポート終了** DDE プロトコルは、16 ビットの Windows 3.x から同じコンピュータ上にある Windows 95/98 データベース・サーバに接続するのに使用されてきましたが、旧バージョンで使用していた Windows 3.x アプリケーションは TCP/IP を使ってバージョン 7.0 データベース・サーバと通信できるので、現在は DDE プロトコルを使用する必要がありません。
- **旧式になった IPX プロトコル** IPX による通信は現在のリリースではまだサポートされていますが、SPX プロトコルを使用することをおすすめします。SPX のプロトコル・オプションは IPX と同じで、パフォーマンスが向上します。IPX については、将来のリリースではサポートが打ち切られる予定です。

  デフォルトでは、データベース・サーバとクライアント・ソフトウェアの両方で IPX プロトコルを使用するには、-x オプションまたは **CommLinks** 接続パラメータを使用して明示的に指定する必要があります。デフォルトでは、SPX プロトコルが使用されます。

  クライアント側から SPX を使用する方法については、「CommLinks 接続パラメータ [LINKS]」『SQL Anywhere サーバ - データベース管理』を参照してください。サーバ側から SPX を使用する方法については、「-x サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **旧式になったネットワーク・プロトコル・オプション** Broadcast と CommAutoStop プロトコル・オプションは使用可能ですが、効果はありません。Adaptive Server Anywhere の将来のバージョンではサポートが打ち切られる予定です。
- **dbclient 互換性実行プログラム廃止** バージョン 6 では、dbcli6.exe ユーティリティを使い、簡単にバージョン 5 クライアント接続方式と互換性を持つことができました。バージョン 7 には、これと同等のユーティリティはありません。

動作の変更

このリストには、既存の機能の中で動作に変更があり、アプリケーションに影響のあるもの、または開発やデータベース管理に影響のあるものがまとめられています。

- **Interactive SQL の変更** Interactive SQL の新バージョンには、前バージョンからいくつか変更になった点があります。もともとこれは対話型のツールなので、説明が必要な変更はほとんどありません。

  INPUT 文と OUTPUT 文で、サポートするフォーマットが変更になり、以下のフォーマットがサポートされるようになりました。

  - **INPUT** ASCII、DBASE、DBASEII、DBASEIII、EXCEL、FIXED、FOXPRO、LOTUS
  - **OUTPUT** ASCII、DBASE、DBASEII、DBASEIII、EXCEL、FIXED、FOXPRO、HTML、LOTUS、SQL

- **サーバのネーム・スペース変更** ネットワーク上で、TCP/IP を使用して同じ名前を持つ複数のデータベース・サーバを実行できなくなりました。以前は、ポートが違えば、同じ名前の複数のサーバを実行できました。

- **delete_old_logs が On の場合、ミラーリングされたログは削除** これまで、古いトランザクション・ログのプライマリ・コピーが削除されても、それからミラーリングされたトランザクション・ログは削除されませんでした。

- **ODBC SQLDescribeCol の動作** **@@identity** フィールドでの **SQLDescribeCol** 呼び出しは、SQL_BIGINT を返すようになりました。これまでのバージョンでは、SQL_INTEGER が返されました。

- **更新の制約** 新しく ansi_update_constraints オプションが追加されました。このオプションを Cursors または Strict に設定すると、更新対象が ANSI 標準に準拠するものに制約されます。このオプションを Off に設定すると、従来の動作と同じで、更新可能な範囲が広くなります。

  詳細については、「ansi_update_constraints オプション [互換性]」『SQL Anywhere サーバ - データベース管理』と「UPDATE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **識別子の長さ制限** 長い識別子の取り扱い方がこれまでよりも整合性を持つようになりました。128 バイトを超える識別子は、名前を指定するデータベース・オブジェクトの型によって、許可されることもされないこともありましたが、現在は、128 バイトを超える識別子を定義しようとすると、すべてエラーが返されます。

  詳細については、「識別子」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **jConnect 接続** jConnect 経由で Adaptive Server Anywhere データベース・サーバの名前付きデータベースに接続するのに REMOTEPWD フィールドを使用する場合、本ソフトウェア付属の jConnect バージョン 4.2 以上では、別の形式でフィールドを割り当てる必要があります。

  詳細については、「ドライバへの URL の指定」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **ユーザ定義のエラー** プロシージャとトリガ内で、99000 から 99999 までの範囲にユーザ定義のエラーを例外として複合文で宣言できます。これらのエラーは、SIGNAL 文を使用して処理します。

詳細については、「BEGIN 文」『SQL Anywhere サーバ - SQL リファレンス』と「SIGNAL 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **LOAD TABLE と UNLOAD TABLE のセキュリティ** データベース・サーバ・オプションが追加され、LOAD TABLE 文と UNLOAD TABLE 文を実行するのに必要なパーミッションを制御できるようになりました。

  詳細については、「-gl サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **トリガの @@identity** オートインクリメント・カラムを持つテーブル (T1) INSERT トリガがあり、そのトリガが同様にオートインクリメント・カラムを持つ別のテーブル (T2) への挿入を実行する場合、これまでは、挿入が完了した後、T1 に割り当てたオートインクリメント値の取得はできませんでした。その時点で、@@identity の値は、T2 に割り当てた値になりました。しかし、@@identity の動作が変更になり、値にアクセスできるようになりました。

  トリガ内の @@identity の新しい動作については、「@@identity グローバル変数」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **Embedded SQL DECL_FIXCHAR** これまでのリリースでは、SQL プリプロセッサが DECL_FIXCHAR 型を配列に変換していました。たとえば、DECL_FIXCHAR(12)name; は、char name[12]; に変換されていました。

  現在のリリースでは、SQL プリプロセッサが DECL_FIXCHAR 宣言を "array" という char 配列のメンバ 1 つからなる構造体を使用した宣言に変換します。たとえば、DECL_FIXCHAR (12)name; は、struct {char array[12];} name; に変換されます。前述のように宣言された変数内の個々の文字への参照は、name.array[i] という形式になります。

SQL Remote の動作の変更

次の動作がバージョン 7.0 で変更されました。

- **dbxtract は内部アンロードを使用する** dbxtract はデフォルトで、OUTPUT 文を使用してクライアント側のデータをアンロードする代わりに、UNLOAD 文を使用してサーバ側のデータをアンロードします。従来のリリースの -i と -x のオプションの代わりに -ii、-ix、-xi、-xx オプションを使うと、使用する内部オペレーションと外部オペレーションの組み合わせを選択できます。

  オプションの完全なリストについては、「抽出ユーティリティ」『SQL Remote』を参照してください。

Mobile Link と Ultra Light の動作の変更

- **新しいテーブル名とスクリプト名** 同期スクリプトに関連する情報が含まれている統合データベースのテーブルに、新しい名前が付きました。以前は、こうしたテーブルの名前は、プレフィクス **ul_** で始まっていました。このプレフィクスが、**ml_** に変わりました。古い統合デー

タベースは、バージョン 7.0 との互換性を持たせるためにアップグレードする必要があります。

同様に、テーブル・スクリプトの追加を容易にするストアド・プロシージャの名前が、**sp_table_script** から **ml_add_table_script** に変更されました。また、接続スクリプトの追加を容易にするストアド・プロシージャの名前が、**sp_connection_script** から **ml_add_connection_script** に変更されました。

DB2 上では、これらの名前は 18 文字にトランケートされます。

- **バージョン名を必要とする同期スクリプト**　同期スクリプトには、スクリプト・バージョン名の割り当てが必要になりました。スクリプト・バージョン名によって、Mobile Link サーバは異なるクライアントを別々に処理できます。

第 13 章

バージョン 6.0.3 の新機能

目次

バージョン 6.0.3 の新機能

この項では、SQL Anywhere Studio バージョン 6.0.3 のコンポーネントに導入された新機能について説明します。

Adaptive Server Anywhere の新機能

Adaptive Server Anywhere バージョン 6.0.3 では、バグの修正に加え、ソフトウェアとマニュアルに新しい機能が組み込まれました。

- **ストアド・プロシージャと Java デバッガ** 従来のバージョンで提供された Java デバッガが、アップグレードされました。新しいバージョンのデバッガは、データベース内の Java クラスだけでなく、SQL ストアド・プロシージャとトリガもデバッグできるようになりました。

  デバッガの使用方法については、「プロシージャ、関数、トリガ、イベントのデバッグ」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **読み込み専用データベース** データベース・サーバを起動したときに、データベースを読み込み専用に指定できます。この機能により、CD-ROM などの読み込み専用メディアにデータベースを容易に展開できます。

  読み込み専用データベースの場合、**ReadOnly** データベース・プロパティは On を返し、読み込み専用モードで実行されていないデータベースの場合は Off を返します。

  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」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **計算カラムの拡張** 計算カラムを、これまで以上に柔軟に使用できるようになりました。空でないテーブルに計算カラムを追加し、計算カラムと対応する式を変更できます。計算カラムはいくつかの条件下で計算し直され、常に信頼性の高い値を保持します。

  詳細については、「計算カラムの使用」『SQL Anywhere サーバ - SQL の使用法』と「計算カラムの挿入と更新」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

  構文の詳細については、「ALTER TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **ユーロ通貨のサポート** ユーロ通貨記号を含む照合が追加されました。これらは、1252LATIN1 と ISO9LATIN1 照合です。

- **照合の追加** 852POL (OEM Code Page 852 ラテン文字 2、ポーランド語順)、1250POL (Windows Latin2 Code Page 1250、ポーランド語順)、1250Latin2 (Windows Latin2 Code Page 1250)、932JPN (日本語)、936ZHO (EUC_CHINA に類似した言語)、950TAI (EUC_TAIWAN に類似した言語) など、これまでになかった照合が、提供される照合のリストに追加されました。

  完全なリストについては、「照合の選択」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しい Windows CE プラットフォーム**　SH4 と ARM プロセッサが、Windows CE 2.1x でサポートされるようになりました。
- **ALTER TABLE の拡張**　ALTER TABLE 文が拡張され、カラムにデフォルト値の設定と削除を行う、SQL/92 と互換性のある句を提供できるようになりました。これらの句は、既存の MODIFY 句の代わりに使用できます。

  ```
  ALTER column-name SET DEFAULT default-value
  | ALTER column-name DROP DEFAULT
  ```

  詳細については、「ALTER TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **LOAD TABLE の拡張**　LOAD TABLE 文を使ってテーブルの指定したカラムをロードできるようになりました。また、再構築に関する問題を処理する新しい CHECK CONSTRAINTS オプションが追加されました。

  詳細については、「LOAD TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **ファイアウォール経由の容易な接続**　ファイアウォールの内外の接続を容易にするプロトコル・オプションのセットが組み込まれました。

  詳細については、「ファイアウォール経由の接続」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **BACKUP 文の拡張**　トランザクション・ログのバックアップ・コピーの名前を *YYMMDDnn.log* という形式のファイル名に変更できる MATCH キーワードが追加されました。このキーワードを使用すれば、データを上書きせずに同じ文を複数回実行できます。

  詳細については、「BACKUP 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **容易なアンロードと再ロード**　アンロード [dbunload] ユーティリティが強化され (-ar オプション)、データベースがレプリケーションの対象であるかどうかにかかわらず、使用できるデータベースのアンロードと再ロードを 1 回の操作でできるようになりました。

  詳細については、「アンロード・ユーティリティ (dbunload)」『SQL Anywhere サーバ - データベース管理』と「データベースの再構築」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **テンポラリ・ファイルのロケーション**　テンポラリ・ファイルのロケーションを判断するとき、データベース・サーバは新しい環境変数 ASTMP をチェックします。これにより、システムのテンポラリ・ディレクトリ以外のディレクトリにテンポラリ・ファイルを置くことができます。

  詳細については、「SATMP 環境変数」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しいシステム・プロシージャ**　新しいシステム・プロシージャを使用すると、DBA ユーザはデータベース・サーバのオプション (sa_server_option) を無効にし、データベース・サーバ・キャッシュ (sa_flush_cache) をフラッシュできます。

  詳細については、「システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **文字セット変換のチューニング**　新しい CharSet 接続パラメータを使用して、個々の接続に対し、文字セット変換で使用するアプリケーション・ロケールを制御できるようになりました。

  詳細については、「CharSet 接続パラメータ [CS]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **再編成されたパフォーマンス・モニタの統計**　Windows NT パフォーマンス・モニタで使用できる統計が、エリアに編成されました。いくつかの統計が追加され、利用頻度の少なかった統計は削除されました。

  使用可能な統計のリストについては、「Windows パフォーマンス・モニタを使用した統計値のモニタリング」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **ユーティリティ・データベースのデータベース・プロパティ**　ユーティリティ・データベースに対して、テーブル指定のない SELECT 文を実行できるようになりました。この機能は、主にデータベースと接続プロパティの検索に使用されます。

  詳細については、「ユーティリティ・データベースの使用」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **新しいデータベース・プロパティ**　プロパティ関数を使用して次のプロパティを利用できます。

  - **IsNetworkServer**　ネットワーク・データベース・サーバに接続されると YES を返し、パーソナル・データベース・サーバに接続されると NO を返します。

    詳細については、「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

  - **DefaultCollation**　新しい **DefaultCollation** プロパティを使用して、データベースを作成するときに使用するデフォルト照合を検索できます。

    詳細については、「デフォルトの照合の判断」『SQL Anywhere サーバ - データベース管理』と「サーバ・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

  - **MultiByteCharSet**　**MultiByteCharSet** データベース・プロパティを使用すると、データベースがマルチバイト照合またはシングルバイト照合のどちらを使用しているかをチェックできます。

    このプロパティの詳細については、「データベース・レベルのプロパティ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **内部 JDBC における JDBC 2.0 関数のサポート**　内部サーバ側 JDBC ドライバは、JDBC 2.0 インタフェースからの関数をサポートするようになりました。サーバ側の Java アプリケーションは、スクロール可能で更新可能な結果セットやバッチ更新などの機能を使用できるようになりました。これに伴い、Interactive SQL から Java メソッドによる結果セットにアクセスできるようになりました。

- **Java クラスで main メソッドを使用する**　SQL から Java クラスの main メソッドを実行できます。

  詳細については、「main メソッドの呼び出し」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **ユーザ定義関数に Java クラスを使用する**　SQL ユーザ定義関数に、Java メソッドを含めることができます。

  詳細については、「CREATE FUNCTION 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **Java メソッドを使用するストアド・プロシージャの拡張**　Java メソッドに含まれるストアド・プロシージャに、OUT と INOUT パラメータを使用できます。

  詳細については、「Java からストアド・プロシージャを経由して値を返す」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **データベース内のマルチスレッド Java クラス**　java.lang.thread パッケージへのサポートが追加されました。

  詳細については、「Java アプリケーションでのスレッドの使用」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **Java からファイルへのアクセス**　データベースのクラスからファイルへのアクセスを可能にするクラスを含め、*java.io* パッケージのすべてのクラスに対するサポートが追加されました。java_input_output オプションが導入されましたが、セキュリティの理由により、この機能は DBA だけが設定できるようになっています。

  この機能は Windows NT と UNIX でのみ提供されます。

- **CONVERT 関数の拡張**　CONVERT 関数がサポートする日付と時刻のスタイルが拡張されました。

  詳細については、「CONVERT 関数 [データ型変換]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **データベース・サーバの起動ダイアログ**　32 ビットの Windows オペレーティング・システムでは、データベース・サーバを引数なしで開始すると、データベース・ファイルと追加のパラメータを指定するウィンドウが表示されます。

  詳細については、「データベース・サーバの起動」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **ログ変換 [dbtran] ユーティリティの強化** ログ変換 [dbtran] ユーティリティを使うと、トランザクション・ログ・オペレーションのフィルタリングによりオペレーションのサブセットを分離できます。

  詳細については、「ログ変換ユーティリティ (dbtran)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **トランザクション・ログ [dblog] ユーティリティの強化** トランザクション・ログ [dblog] ユーティリティを使用して、オフセット情報を含めて追加のサマリ情報が表示されるようになりました。

  詳細については、「トランザクション・ログ・ユーティリティ (dblog)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **開始サーバ・イン・バックグラウンド・ユーティリティの強化** 開始サーバ・イン・バックグラウンド・ユーティリティ (dbspawn) に -f オプションが追加され、すでに実行中のサーバがあっても、サーバを強制的に起動できるようになりました。このオプションは、db_start_engine Embedded SQL 関数だけで使用され、ForceStart 接続パラメータを指定します。

  詳細については、「サーバ・バックグラウンド起動ユーティリティ (dbspawn)」『SQL Anywhere サーバ - データベース管理』と「db_start_engine 関数」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **Replication Agent をデーモンとして実行する** UNIX オペレーティング・システムでは、-ud オプションを指定することによって、Replication Agent をデーモンとして実行できます。

  詳細については、「Log Transfer Manager ユーティリティ (dbltm)」『SQL Anywhere サーバ - データベース管理』を参照してください。

SQL Remote の新機能

SQL Remote バージョン 6.0.3 では、バグの修正に加え、次の新しい機能が追加されました。このリストには、Adaptive Server Anywhere の機能で特に SQL Remote に関係のあるものも含まれています。

- **UNIX による FTP と SMTP/POP のサポート** UNIX オペレーティング・システムでサポートされるメッセージ・システムの範囲が拡張され、FTP と SMTP/POP もサポートされるようになりました。

- **データベースに格納されるメッセージ・リンクのオプション** 各メッセージ・システムで SQL Remote の動作を制御するメッセージ・リンク・パラメータは、レジストリではなく、データベースに格納されるようになりました。これによって、メッセージ・リンク・パラメータに関連する配備と管理が簡略化されました。

- **日付と時間のレプリケーション・フォーマット** データベース・オプションを指定して、日付と時間をレプリケートするときに使用するフォーマットを SQL Remote に指示できるようになりました。使用するオプションは sr_time_format、sr_date_format、sr_timestamp_format です。

詳細については、「日付と時刻のレプリケーション」『SQL Remote』と「SQL Remote オプション」『SQL Remote』を参照してください。

- **Message Agent と SQL Remote Open Server をデーモンとして実行する** UNIX オペレーティング・システムでは、-ud オプションを使用して、この 2 つのアプリケーションをデーモンとして実行できます。
- **Adaptive Server Anywhere データベースの容易なアンロードと再ロード** アンロード [dbunload] ユーティリティが強化され (-ar オプション)、データベースがレプリケーションの対象であるかどうかにかかわらず、使用できるデータベースのアンロードと再ロードを 1 回の操作でできるようになりました。

  詳細については、「アンロード・ユーティリティ (dbunload)」『SQL Anywhere サーバ - データベース管理』と「データベースの再構築」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **トランザクション・ログ [dblog] 出力の強化** トランザクション・ログ [dblog] ユーティリティを使用して、オフセット情報を含めて追加のサマリ情報が表示されるようになりました。

  詳細については、「トランザクション・ログ・ユーティリティ (dblog)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **ログ変換 [dbtran] ユーティリティの強化** ログ変換 [dbtran] ユーティリティを使うと、トランザクション・ログ・オペレーションのフィルタリングによりオペレーションのサブセットを分離できます。この機能を使用できるのは、SQL Remote の管理者だけです。

  詳細については、「ログ変換ユーティリティ (dbtran)」『SQL Anywhere サーバ - データベース管理』を参照してください。

Mobile Link と Ultra Light の新機能

バージョン 6.0.2 以降のソフトウェアへの変更点と追加点を次に示します。

- **新しいデータ型** real データ型と double データ型が完全にサポートされるようになりました。
- **文字セット変換** Mobile Link サーバは、アップロードされたすべての文字をユニコードに変換し、ユニコード ODBC API を使用して統合データベースに渡すようになりました。逆に、ダウンロードされたすべての文字をユニコードから Ultra Light アプリケーションの文字セットに変換します。統合データベース・サーバでの文字セット変換が結果に影響を及ぼすことがありますが、新しいシステムでは、複数のプラットフォームでより一貫した動作を行えます。

  詳細については、「ODBC ドライバの文字セット変換の制御」『Mobile Link - サーバ管理』を参照してください。
- **Windows NT サービスとして実行する Mobile Link サーバ** Mobile Link サーバをサービスとして実行する場合は、Windows NT ワークステーションをログオフしても実行し続けるように設定できます。

- **DB2 設定スクリプトの提供**　IBM DB2 を統合データベースとしてより簡単に使用できるようにするため、使用可能なセット・スクリプトに DB2 設定スクリプトが追加されました。

  設定スクリプトのリストについては、「統合データベースの設定」『Mobile Link - サーバ管理』を参照してください。

バージョン 6.0.3 での動作の変更

この項では、SQL Anywhere Studio バージョン 6.0.3 のコンポーネントに導入された動作の変更について説明します。

Adaptive Server Anywhere の動作の変更

- **追加カラムにデフォルト値を設定する** デフォルト値のあるカラムが追加された場合、その追加カラム全体にデフォルト値が設定されます。従来のリリースでは、カラムに NULL が設定されました。
- **参照整合性アクションのパーミッション** プライマリ・テーブルが変更されると、カスケード削除や更新などの参照整合性アクションがセカンダリ・テーブルで実行されます。参照整合性アクションは、システム・トリガを使って行われます。このトリガは、セカンダリ・テーブルの所有者のパーミッションを取得して実行されるようになりました。以前は、プライマリ・テーブルの所有者のパーミッションを取得してトリガが実行されていました。新しい動作では、カスケード・オペレーションを異なる所有者のテーブル間で実行するのに、追加のパーミッションを取得する必要はありません。
- **datediff、MONTHS、YEARS 関数** 2 つの日付間の月数は、2 つの日付の間にある月の第 1 日目の数として計算されます。たとえば、1 月 25 日と 2 月 2 日の差異は 1 (ヶ月) に、1 月 1 日と 1 月 31 日の差異は 0 (ヶ月) になります。年数は、2 つの日付の間に存在する 1 月 1 日の数として計算されるようになりました。

  これにより、新旧の関数で結果として得られる数字が 1 異なることがあります。この変更は、Adaptive Server Enterprise との互換性のために行われました。

  より短い時間単位の場合は、DATEDIFF 関数に対するオーバフロー値が使われるようになりました。従来のバージョンでは、制限を越えると間違った答がでました。

  詳細については、「DATEDIFF 関数 [日付と時刻]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **デフォルトのページ・サイズ** データベースのデフォルト・ページ・サイズが 2048 バイトになりました。これは、多くのユーザにとって便利です。
- **デフォルト・データベース照合** データベースを作成するときに使うデフォルト照合が変更されました。このデフォルト照合は使用するオペレーティング・システムの設定によって異なります。

  デフォルト照合の検索方法については、「デフォルトの照合の判断」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **SQL プリプロセッサのデフォルト照合** 照合を明示的に指定しない場合、Embedded SQL プリプロセッサは、ロケール情報を使用してデフォルト照合を選択するようになりました。ロケール情報がない場合は、850LATIN1 を使用します。使用する照合はバナーの後にレポートされます。従来の動作では 850 を使用しました。

プリプロセッサの詳細については、「SQL プリプロセッサ」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **サーバ名の長さの強制適用** サーバ名は起動時にチェックされ、最長 40 文字にトランケートされます。NetBIOS では、16 文字にトランケートされます。クライアント側では、EngineName パラメータの値も 40 文字にトランケートされます。

  詳細については、「EngineName 接続パラメータ [ENG]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Agent 接続パラメータ** Agent 接続パラメータの動作が変更されました。このパラメータの意味がバージョン 5 からバージョン 6 で変更されたのは、*dbclient* 実行プログラムが不要になったためです。パラメータの意味が変更されて、バージョン 6 の環境でより便利になりました。

  Agent 接続パラメータは、バージョン 8.0.1 からは使用されなくなりました。

SQL Remote の動作の変更

SQL Remote バージョン 6.0.3 では、次の動作が変更されました。

- **データベースに格納されるメッセージ・リンク・パラメータ** デフォルトでは、新しいバージョンのソフトウェアで最初に Message Agent を実行するときに、メッセージ・リンク・パラメータがデータベースに移動します。データベースの外部にある変更前のロケーションで、これらのパラメータを明示的に探すソフトウェアを使用している場合には、エラーが発生します。external_remote_options データベース・オプションを On に設定すると、従来のバージョンどおりの動作となります。

- **パスワードの格納** メッセージ・リンクのパスワードを入力したときに、従来のバージョンのソフトウェアでは、パスワードが格納されませんでした。新しいバージョンではパラメータがデータベースに保管されるようになったため、保存されたパスワードはディスク上に格納されず、セキュリティが向上しました。パスワードはデフォルトで保存されるようになりました。save_remote_passwords オプションを OFF に設定すると、従来のバージョンと同じ動作となります。

第 14 章

バージョン 6.0.2 の新機能

目次

バージョン 6.0.2 の新機能

この項では、SQL Anywhere Studio バージョン 6.0.2 のコンポーネントに導入された新機能について説明します。

Adaptive Server Anywhere の新機能

Adaptive Server Anywhere バージョン 6.0.2 では、バグの修正に加え、ソフトウェアとマニュアルに新しい機能が組み込まれました。

相互参照
製本版のマニュアルは、メンテナンス・リリースごとに更新されているわけではありません。この項の相互参照は製本版のマニュアルでは有効でない場合があります。最新の情報については、オンライン・マニュアル (英語版) を参照してください。

- **Ultra Light 展開オプション**　このバージョンでは、PalmPilot や Windows CE などの小型デバイス用 Ultra Light データベースを開発できます。

  詳細については、『*Ultra Light 開発者用ガイド*』を参照してください。

- **SQL 文のバックアップとリストア**　BACKUP と RESTORE を SQL 文として追加することにより、SQL スクリプトを使用したサーバ側のバックアップとバックアップの自動化が可能になります。

  BACKUP 文によって、テープへの直接バックアップが可能になります。

  詳細については、「BACKUP 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **セキュリティ機能**　新しいセキュリティ機能が追加されました。

  - **監査**　データベース管理者は、auditing オプションをオンにすることで、データベースで実行されたアクティビティを追跡できます。アクティビティのレコードはトランザクション・ログに保持されます。監査をオンにすると、ログイン試行、すべてのイベントの正確なタイムスタンプ、すべてのパーミッション・チェック、DBA 権限を必要とするすべてのアクションを含むため、トランザクション・ログに保存されるデータ量が増加します。

    詳細については、「データベース・アクティビティの監査」『SQL Anywhere サーバ - データベース管理』を参照してください。

  - **パスワードの最小長**　データベース管理者は、パスワードが簡単に解読されないように、パスワードの最小長を指定できます。

    詳細については、「min_password_length オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **サーバの検出**　接続のトラブルシューティング用にユーティリティが提供されています。

詳細については、「Ping ユーティリティ (dbping)」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **jConnect 接続によるデータベースの開始** jConnect を介した Java アプリケーションからの接続を含む TDS データベース接続を実行すると、サーバ上でデータベースを起動できます。

  詳細については、「ドライバへの URL の指定」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **ODBC 3.51** ODBC ドライバは ODBC 3.51 に更新されました。このバージョンの ODBC ではユニコード・アプリケーションもサポートしています。

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

- **使用できる JOIN 構文の制御** これまでのリリースでは、あいまいなジョイン句を持つマルチテーブル・クエリが許可されていましたが、今回のリリースでは、オプションを設定してこのようなクエリを許可しないようにできます。

  詳細については、「extended_join_syntax オプション [データベース]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **管理ユーティリティの強化** 新しい機能を提供するオプションが、管理ユーティリティに追加されました。

  - **トランザクション・ログ [dbtran] ユーティリティ** 新しい -d オプションを使用すると、dbtran は、個々のオペレーションを、その発生時にトランザクション・ログ・ファイルに記録します。これによりトランザクション・ログの出力が読みやすくなります。これは主に監査を目的として追加されました。

    詳細については、「ログ変換ユーティリティ (dbtran)」『SQL Anywhere サーバ - データベース管理』を参照してください。

  - **トランザクション・ログ [dbtran] ユーティリティ** dbtran は、ログ・ファイルではなく、稼働中のデータベース・サーバに対して実行できます。この機能を使用するとトランザクション・ログに直接アクセスする必要がないので、トランザクション・ログのセキュリティを強化するために追加されました。

    詳細については、「ログ変換ユーティリティ (dbtran)」『SQL Anywhere サーバ - データベース管理』を参照してください。

  - **Log Transfer Manager [dbltm] ユーティリティのロギング** メッセージのロギングをチューニングするための新しいオプションが追加されました。

    詳細については、「Log Transfer Manager ユーティリティ (dbltm)」『SQL Anywhere サーバ - データベース管理』を参照してください。

  - **新しい Log Transfer Manager [dbltm] ユーティリティのオプション** 新しいオプションとして、バックアップされたトランザクションのみをレプリケートするオプション (**backup_only**) と、すべてのデータのレプリケート完了とともに停止するオプション (**continuous**) が加わりました。

詳細については、「LTM 設定ファイル」 『SQL Anywhere サーバ - データベース管理』を参照してください。

SQL Remote の新機能

SQL Remote バージョン 6.0.2 では、バグの修正に加え、次の新しい機能が追加されました。

- **パフォーマンスの強化** トランザクション・ログのスキャンとメッセージの送信を行うための Adaptive Server Anywhere Message Agent (dbremote) オペレーショナル・モデルの強化により、レプリケーションのターンアラウンド・タイムが大幅に短縮されました。

  あるサイトにデータを入力してから、そのレプリケーションを別のサイトに入力するまでの最小遅延時間は、従来のバージョンではおよそ 10 分という制約がありました。新しいオペレーショナル・モデルでは、状況によって最小遅延時間を秒単位で設定することができます。

  Message Agent のメッセージ送信処理は、継続モードで実行されている場合、必要なデータがコミットされるのを待っている間アクティブなトランザクション・ログの終わりに置かれ(「**ホーバー**」状態)、そのたびにトランザクション・ログをスキャンし直しません。これによってより頻繁なポーリングが可能になり、レプリケーションにかかる時間が大幅に削減されます。

  詳細については、「Message Agent のパフォーマンスのチューニング」 『SQL Remote』を参照してください。

- **SQL Remote メッセージのロギング** メッセージのロギングをチューニングするための新しいオプションが追加されました。

  詳細については、「Message Agent」 『SQL Remote』を参照してください。

バージョン 6.0.2 での動作の変更

この項では、SQL Anywhere Studio のコンポーネントでの動作の変更について説明します。

Adaptive Server Anywhere の動作の変更

ここでは、これまでのバージョンとは異なる動作について説明します。

- **Java のデバッグに必要なパーミッション** Java デバッガを使用するには、DBA 権限を持っているか、SA_DEBUG グループのメンバシップを付与されている必要があります。SA_DEBUG グループは、6.0.2 より前に作成されたデータベースには存在しません。このような古いデータベースでは、任意のユーザが Java デバッガを使用できます。SA_DEBUG グループは、セキュリティ上のループ・ホールを防止するために追加されました。

  詳細については、「Java デバッガの稼働条件」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **デフォルト・パケット・サイズの変更** クライアント／サーバ通信のデフォルト・パケット・サイズが、512 バイトから 1000 バイトに変更されました。この変更によって、マルチロー・フェッチと大きなローのフェッチのパフォーマンスが向上します。同時に必要なメモリも増加します。

  パケット・サイズの詳細については、「CommBufferSize 接続パラメータ [CBSIZE]」『SQL Anywhere サーバ - データベース管理』と「-p サーバ・オプション」『SQL Anywhere サーバ - データベース管理』を参照してください。

第 15 章

バージョン 6.0.1 の新機能

目次

バージョン 6.0.1 の新機能

この項では、Adaptive Server Anywhere バージョン 6.0.1 の新機能について説明します。新機能の一覧を挙げ、各機能の詳細が説明されている参照先も示します。

Adaptive Server Anywhere for Windows CE

Microsoft Windows CE は、ハンドヘルド・コンピュータ・デバイス用と特定のタスクを実行するために特別に開発された埋め込みデバイス用オペレーティング・システムです。

バージョン 6.0.1 以降の Adaptive Server Anywhere は、Windows CE でも使用できます。Adaptive Server Anywhere Windows CE 版には、次の特長があります。

- **完全な機能を備えたデータベース**　Adaptive Server Anywhere の他のバージョンが持つすべての SQL 機能を、Windows CE 版でも使用できます。これらの機能には、トランザクション処理、参照整合性の確保、プロシージャとトリガなどが含まれます。

  Java 機能とリモート・データ・アクセス機能は、Windows CE では使用できません。

- **デスクトップからの管理**　PC へのネットワーク接続または直接接続が可能なデバイスで Windows CE を実行している場合は、PC 上で実行している Sybase Central から Windows CE データベースを管理できます。

- **ODBC と Embedded SQL アプリケーション**　クライアント・アプリケーションの開発には、これらのインタフェースのどちらでも使用できます。

- **SQL Remote レプリケーション**　Windows CE ActiveSync による同期との互換性を保つために、SQL Remote ファイル・リンクが実装されています。

リモート・データ・アクセス

リモート・データ・アクセス機能は、外部データ・ソースがあたかもローカル・データベース上に保存されているかのようにアクセスすることを可能にします。

リモート・データ・アクセスの詳細については、「リモート・データへのアクセス」『SQL Anywhere サーバ - SQL の使用法』と「リモート・データ・アクセスのサーバ・クラス」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

文字セット変換

文字セット変換が追加され、クライアント・アプリケーションとデータベース・サーバとの間で文字セットが渡されるときに、異なる文字セット間で自動的に文字列が変換されるようになりました。これにより、混合文字セット環境での柔軟性が高くなりました。

文字セット変換は、同じ文字を異なる値で表す文字セット間でも実行できます。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。

文字セット変換を有効にするには、新しい -ct オプションを使用して、データベース・サーバを起動してください。

多くの場合、文字セット変換機能は自動的に実行され、ユーザの介入はほとんど必要ありません。

文字セット変換の詳細については、「文字セット変換」『SQL Anywhere サーバ - データベース管理』を参照してください。

新しい Java 機能

Java サポートにはいくつかの変更が加えられました。拡張機能には次のようなものがあります。

- **圧縮 jar ファイル** 圧縮された jar ファイルと zip ファイルをデータベースにインストールできるようになりました。ただし、Sun JDK に付属の *jar* ユーティリティは使用しないでください。それ以外の zip ユーティリティを使用すれば適切なファイルが生成されます。
- **Java プロシージャからの結果セット** 1 つのストアド・プロシージャに Java メソッドをラップできます。これによって、呼び出し元の環境に 1 つの結果セット、または複数の結果セットを返すことができます。

  この機能の詳細については、「Java メソッドから返される結果セット」『SQL Anywhere サーバ - プログラミング』を参照してください。
- **デフォルトの内部接続** 内部 JDBC オペレーション用にデータベース接続を確立する場合、次の URL の使用をおすすめします。

  ```
  jdbc:default:connection
  ```

  バージョン 6.0.0 では、この接続を確立するのに空の文字列を使用していました。新バージョンでも空の文字列を使用できますが、優先されません。SQLJ1 が推奨する標準に対応する新しい URL をお使いください。

追加の新機能

Adaptive Server Anywhere バージョン 6.0.1 のその他の新しい機能には、以下のものが含まれます。

- **jConnect 4.0** この製品に含まれる jConnect が、バージョン 4.0 に更新されました。
- **自動スタート接続パラメータ** このパラメータを使用すると、ネットワーク接続が成功しなかった場合にはパーソナル・サーバを起動しないように設定できます。

  詳細については、「AutoStart 接続パラメータ [ASTART]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **MESSAGE 文**　MESSAGE 文の拡張機能によって、メッセージをクライアント、サーバ・メッセージ・ウィンドウ、またはログ・ファイルに送信できます。

  詳細については、「MESSAGE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **メッセージ・コールバック**　Windows Embedded SQL アプリケーションは、メッセージ・コールバック関数を登録することによって、要求が処理されている間でもサーバから受信したメッセージを処理できます。

  詳細については、「要求管理の実装」『SQL Anywhere サーバ - プログラミング』を参照してください。

- **オペレーティング・システムのスレッド制御の強化**　新しいデータベース・サーバのオプション (-gx) は、使用中のオペレーティング・システムのスレッド数を制御します。既存の -gt オプションは、同時に使用できるスレッド数を制御し、実際に使用できる CPU の数を制御します。

  詳細については、「SQL Anywhere データベース・サーバ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **接続プロパティ・システム・プロシージャ**　sa_conn_properties_by_conn および sa_conn_properties_by_name システム・プロシージャは、接続情報のクエリに使用できます。

- **NULLIF 関数**　CASE 式の省略形を提供します。NULLIF は 2 つの式の値を比較します。1 番目の式と 2 番目の式が等しい場合、NULLIF は NULL を返します。1 番目の式と 2 番目の式が異なる場合、NULLIF は 1 番目の式を返します。NULLIF 関数は、いくつかの CASE 式の簡単な作成方法を提供します。

  詳細については、「その他の関数」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

SQL Remote の新機能

いくつかの機能が SQL Remote に追加されました。

- **最小化された Message Agent** -q オプションを使用すると、Message Agent を最小化されたウィンドウで起動できます。
- **Message Agent のメッセージの再送信要求** Message Agent が、失われたメッセージをどの時点で再送信するように要求するかは、-rp オプションを使用してユーザが設定できるようになりました。

  これらのオプションの詳細については、「Message Agent」『SQL Remote』と「受信メッセージのポーリングのチューニング」『SQL Remote』を参照してください。
- **ステーブル・キューのクリーニング** Adaptive Server Enterprise では、Message Agent の新しい -fq オプションを使用すると、確認済みのメッセージをステーブル・キューからクリーニングでき、管理しやすくなります。

  詳細については、「Message Agent」『SQL Remote』を参照してください。

動作の変更

ここでは、バージョン 6.0.0 と 6.0.1 とで異なる動作について説明します。

Java システム・テーブルの変更　Java クラス情報を記録するために使用していたシステム・テーブル (SYSJAR、SYSJARCOMPONENT、SYSJAVACLASS) には、SMALLINT プライマリ・キーがありました。これらのデータ型は、INTEGER プライマリ・キーを使用するよう変更されました。この変更によって、より多くの Java クラスをデータベースに格納でき、データベースの Java クラスへのより多くの変更が可能になりました。

この変更は、新しいデータベースと、アップグレード・ユーティリティ (dbupgrad) を使用してこれ以降のリリースからアップグレードされたデータベースに対して有効です。

Transact-SQL と jConnect 接続のデフォルトの ansinull 設定　Adaptive Server Enterprise のデフォルト動作に合わせて On に変更されました。

データベース・サーバ -v オプション　6 より前のバージョンでは、このオプションはトランザクション・ログに冗長出力を生成しました。この動作は変更され、現在では、-v はバージョン情報を提供するのに使用されます。

データベース・サーバ -gss オプション　-gs サーバ・オプションはスタック・サイズの設定に使用されていましたが、動作が複雑でした。現在では、-gs オプションは使用せず、-gss によって同じ機能がより簡潔な方法で提供されます。

詳細については、「SQL Anywhere データベース・サーバ」『SQL Anywhere サーバ - データベース管理』を参照してください。

Interactive SQL での文字セット変換　以前は、char_oem_translation オプションを DETECT に設定すると、Interactive SQL はデータベースから照合ラベルをフェッチして、OEM から ANSI への文字セット変換をオンにするかどうかを決定していました。照合ラベルが ANSI コード・ページを示す文字列で開始されている場合、変換はオフになります。そうでない場合、変換はオンになります。オプションを DETECT に設定すると、Interactive SQL により、データベースの照合ラベルとディスプレイ変換設定を示すメッセージがステータス・ウィンドウに表示されます。

新しい動作は次のとおりです。オプションを DETECT に設定すると、Interactive SQL は、サーバから CharacterSet 接続プロパティを取得します。この文字セットは、サーバがこの接続上のすべての文字列の送信に使用するものです。文字セットが ANSI コード・ページを示す場合、OEM から ANSI への変換はオフになります。そうでない場合、変換はオンになります。新しいメッセージが表示され、データベースの照合ラベル、この接続の通信に使用される文字セット、ディスプレイ変換設定が示されます。

第 16 章

SQL Anywhere 10 へのアップグレード

目次

SQL Anywhere のアップグレード

このバージョンのソフトウェアで既存のアプリケーションを使用する前に、動作の変更のリストを確認して、アプリケーションに影響がないかどうかを確認してください。SQL Anywhere 10 - 変更点とアップグレード 1 ページを参照してください。

注意
Adaptive Server Anywhere は、SQL Anywhere に名前が変更されました。この章では、SQL Anywhere はすべてのバージョンの SQL Anywhere を指します。

バージョン 10.0.0 のデータベースのアップグレード

バージョン 10.0.0 からアップグレードするには、アップグレード・ユーティリティを使用するか、データベースを再構築します。「バージョン 10.0.0 のデータベースのアップグレード」 354 ページと「バージョン 10.0.0 のデータベースの再構築」 355 ページを参照してください。

実体化ビュー (Materialized View) が含まれるデータベース
データベース・サーバをアップグレードした後、またはアップグレード後のデータベース・サーバで使用できるようにデータベースを再構築またはアップグレードした後は、データベース内の実体化ビュー (Materialized View) を再表示することをおすすめします。「実体化ビュー (Materialized View) のリフレッシュ」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

バージョン 9 以前のデータベースのアップグレード

SQL Anywhere バージョン 9 以前からバージョン 10.0.1 にアップグレードするには、データベースを再構築します。再構築するには、古いデータベースをアンロードしてから、バージョン 10 の新しいデータベースに再ロードします。バージョン 9 以前のデータベースをロードしようとすると、データベースの開始時にエラーが発生します。既存のデータベースを再構築するには、次の 3 つの方法があります。

- -an (データベースの新規作成) オプションまたは -ar (古いデータベースの置き換え) オプションを指定して、バージョン 10.0.0 のアンロード・ユーティリティ (dbunload) を実行します。「アップロード・ユーティリティを使用したバージョン 9 以前のデータベースの再構築」 363 ページを参照してください。

  注意
  アンロード ユーティリティ (dbunload) には、SQL Anywhere の全バージョンで同じファイル名が使用されています。正しいバージョンを使用していることを確認してください。「ユーティリティの使用」 352 ページを参照してください。

- Sybase Central の [データベース・アンロード] ウィザードを使用します。新しいデータベースを作成するか、既存のデータベースを新しいデータベースに置き換えるか、データベースをファイルにアンロードします。「Sybase Central からのバージョン 9 以前のデータベースの再構築」 362 ページを参照してください。

- 旧バージョンの dbunload を使用してデータベースをアンロードし、*reload.sql* ファイルとバージョン 10.0.0 のデータベース・サーバを使用してデータベースを再ロードします。スキーマを変更する必要がある場合は、この方法を使用してアップグレードすることをおすすめします。スキーマを変更後は、新しいデータベースを初期化して、再構築したスクリプトを適用できます。

Mac OS X のデータベースの再構築
Mac OS X で SQL Anywhere 9.0.2 以前のデータベースをアンロードするには、9.0.2 以前のソフトウェアを使用します。SQL Anywhere 10 ソフトウェアを使用して Mac OS X データベースをアンロードするには、別のプラットフォームでデータベースをアンロードする必要があります。再ロードは、Mac OS X でバージョン 10 のソフトウェアを使用して行うことができます。

アンロードと再ロードのときに (大文字と小文字を「区別する」から「区別しない」に変更するなど) データベースの特性を変更する場合、手順はもう少し複雑になります。詳細については、「データベースの再構築」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

既存のソフトウェアとの互換性

- SQL Anywhere 10 のデータベース・サーバは、バージョン 6.0.0 以降のソフトウェアを使用するクライアント・アプリケーションからの接続をサポートします。バージョン 5 以前のクライアントは、バージョン 10 のデータベース・サーバに接続できません。バージョン 9 以前のクライアントからバージョン 10 のデータベース・サーバに接続した場合は、次の機能を使用できません。
  - Kerberos ログイン
  - Embedded SQL の NCHAR 型
  - ODBC、OLE DB、ADO.NET からのユニコード・データの向上したサポート (たとえば NCHAR を WCHAR カラムとして記述するなど)
  - ODBC を使用するユニコード・アプリケーションや、OLE DB や ADO.NET を使用するあらゆるアプリケーションでの BLOB パフォーマンスの拡張
  - 主に WAN パフォーマンスを向上させるだけでなく、LAN パフォーマンスも若干向上させる強化

共有メモリ接続のための SATMP の設定
バージョン 9 以前でテンポラリ・ファイルの場所指定に使用されていた検索順序は、バージョン 10 のものとは異なります。バージョン 9 以前のクライアントを共有メモリを使用してバージョン 10 のデータベース・サーバに接続する場合、SATMP (バージョン 10) と ASTMP (バージョン 9 以前) 環境変数を設定して、テンポラリ・ファイルのロケーションを指定する必要があります。これらの環境変数を設定しないと、共有メモリ経由での接続の試行は失敗します。

- 現在のバージョンの Sybase Central には、以前のバージョンのデータベースとデータベース・サーバを管理する次の機能が用意されています。

- バージョン 8 以降のサーバで動作するバージョン 8 以降のデータベースを接続して、管理できます。
- バージョン 8 以降のデータベース・サーバで動作するバージョン 5、6、7、8、または 9 のデータベースに接続し、Sybase Central の [データベース・アンロード] ウィザードを使用してデータベースを再構築できます。
- バージョン 7 以前のデータベース・サーバで動作するバージョン 6 以前のデータベースはサポートされません。

注意
Sybase Central や Interactive SQL などのバージョン 9 のクライアント・アプリケーションを使用してバージョン 10.0.0 以降のデータベースに接続している場合は、接続には jConnect ではなく、デフォルトで iAnywhere JDBC ドライバが使用されます。iAnywhere JDBC ドライバは、SQL Anywhere データベースに接続する場合の推奨 JDBC ドライバです。

ユーティリティの使用

SQL Anywhere の複数のバージョンが同じコンピュータにインストールされている場合は、ユーティリティを使用するとき、システムのパスに注意してください。インストールでは、最も新しくインストールされたバージョンの実行プログラムのディレクトリがシステム・パスの最後に追加されるため、ソフトウェアの最新バージョンをインストールしたにもかかわらず、以前にインストールしたバージョンが実行されている場合があります。

たとえば、パス内で Adaptive Server Anywhere バージョン 8 の実行プログラムのディレクトリが SQL Anywhere バージョン 10 の実行プログラムのディレクトリよりも前にある場合に dbinit コマンドを使用すると、バージョン 8 のユーティリティを使用することになり、その結果、バージョン 8 のデータベースが作成されてしまいます。

バージョン 10 のユーティリティが確実に使用されるようにするには、次の 5 つの方法があります。

- SQL Anywhere 10 の実行プログラム・ディレクトリが、バージョン 10 以前の実行プログラム・ディレクトリより前になるようにシステム・パスを変更します。
- コマンドを実行する前に、現在のディレクトリを SQL Anywhere 10 の実行プログラム・ディレクトリに変更します。
- 実行するユーティリティの正確な場所を示すユーティリティ名への完全修飾パスを指定します。
- 正しいバージョンのユーティリティが使用されるように、環境を変更するスクリプトを作成します。
- 旧バージョンのソフトウェアをアンインストールします。

アップグレードのクイック・スタート

旧バージョンのユーザは、次の手順に従って、データベースをバージョン 10 にアップグレードできます。

- **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 (コマンド・ラインの場合)。**

1. データベースをバックアップします。次に例を示します。

   ```
   dbbackup -c "DBF=mydb.db;UID=DBA;PWD=sql" old-db-backup-dir
   ```

   > **注意**
   > データベースのバックアップには、正しいバージョンの dbbackup を使用してください。「ユーティリティの使用」 352 ページを参照してください。

2. 断片化されているドライブでは、データベースのパフォーマンスが低下するので、可能な場合は、新しいデータベースを保存するドライブの断片化を解除してください。

   断片化を解除しない場合の手順については、「断片化を解除しないバージョン 9 以前のデータベースの再構築」 365 ページを参照してください。

3. バージョン 10 の dbunload ユーティリティは、旧バージョンのデータベース・サーバで実行されているデータベースに対して使用できないので、すべての SQL Anywhere と Adaptive Server Anywhere のデータベース・サーバを停止します。次に例を示します。

   ```
   dbstop -c "DBF=mydb.db;UID=DBA;PWD=sql"
   ```

4. 既存のデータベースをアンロードして、新しいバージョン 10 のデータベースに再ロード (再構築) します。次に例を示します。

   ```
   dbunload -c "DBF=mydb.db;UID=DBA;PWD=sql" -an mydb10.db -o dbunload_log_mydb.txt
   ```

5. 新しいデータベースを使用する前に停止して、バックアップします。次に例を示します。

   ```
   dbstop -c "DBF=mydb10.db;UID=DBA;PWD=sql"
   dbbackup -c "DBF=mydb10.db;UID=DBA;PWD=sql" new-db-backup-dir
   ```

参照

- 「アップロード・ユーティリティを使用したバージョン 9 以前のデータベースの再構築」 363 ページ
- 「Sybase Central からのバージョン 9 以前のデータベースの再構築」 362 ページ

アップグレードを行う前の重要な注意事項

どのようなアプリケーションをアップグレードする場合でも、事前に行うべき作業がいくつかあります。SQL Anywhere をアップグレードする場合も同じです。

- **動作の変更のチェック**　記述されている動作変更が既存のアプリケーションに影響を与えないことを確認してください。影響する場合は、問題が起こらないように既存のアプリケーショ

ンを更新してください。SQL Anywhere 10 - 変更点とアップグレード 1 ページを参照してください。

- **アプリケーションのテスト**　事前に SQL Anywhere 10 環境でアプリケーションのテストを徹底的に行ってから、実際の運用に使用するアプリケーションをアップグレードしてください。
- **正しいバージョンのユーティリティの使用**　新しいデータベースには、正しいバージョンのデータベース・ユーティリティを使用してください。「ユーティリティの使用」 352 ページを参照してください。
- **データベースの検証とバックアップ**　データベースを検証し、既存のソフトウェアとデータベースをバックアップします。また、データベースのアップグレード前にさかのぼるリカバリはできないので、アップグレード後にバックアップして、その後のリカバリに備えてください。
- **アップグレードの前に同期**　Ultra Light データベースや、Mobile Link インストール環境の SQL Anywhere リモート・データベースなど、同期に関連するデータベースの場合は、同期を正しく行ってからアップグレードしてください。
- **アップグレード手順のテスト**　アップグレード手順は、十分にテストしてから運用システムで実行してください。

SQL Anywhere はさまざまな設定で使用されるので、すべての場合に対応できるアップグレード手順があるわけではありません。

バージョン 10.0.0 の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と、バージョン 10 の機能を有効にするために、システム・テーブル、システム・プロシージャ、データベース・オプションが追加、変更されます。ディスク上でデータが保管、アクセスされるファイル・フォーマットは変更されないので、ソフトウェアの最新バージョンの新機能とパフォーマンス向上をすべて利用できるわけではありません。

データベースのファイル・フォーマットのアップグレードの詳細については、「バージョン 10.0.0 のデータベースの再構築」 355 ページを参照してください。

- **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 (Sybase Central の場合)。**

1. ソフトウェアをアップグレードする場合の一般的な対応策を行います。「アップグレードを行う前の重要な注意事項」 353 ページを参照してください。
2. Sybase Central を起動します。

   [スタート] – [プログラム] – [SQL Anywhere 10] – [Sybase Central] を選択します。
3. アップグレードするデータベースがあるバージョン 10 のデータベース・サーバを起動します。
4. [ツール] メニューから、[SQL Anywhere 10] – [データベースのアップグレード] を選択します。

   [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。
6. アップグレードしたデータベースを使用する前に、データベースを停止し、トランザクション・ログを圧縮します。

   [データベース・アップグレード] ウィザードの使用の詳細については、「データベースのアップグレード」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

♦ **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 (コマンド・ラインの場合)。**

1. ソフトウェアをアップグレードする場合の一般的な対応策を行います。「アップグレードを行う前の重要な注意事項」 353 ページを参照してください。
2. アップグレードするデータベースに排他的にアクセスできること、またシステム・パスで、バージョン 10 のユーティリティがその他のユーティリティより前に指定されていることを確認します。「ユーティリティの使用」 352 ページを参照してください。
3. データベースに対してアップグレード・ユーティリティ (dbupgrad) を実行します。

   ```
   dbupgrad -c "connection-string"
   ```

   connection-string で指定するデータベース・ユーザは、アンロードするデータベースに DBA 権限で接続する必要があります。

   詳細については、「アップグレード・ユーティリティ (dbupgrad)」 『SQL Anywhere サーバ - データベース管理』を参照してください。
4. アップグレードしたデータベースを使用する前に、データベースを停止し、トランザクション・ログを圧縮します。

♦ **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 (SQL の場合)。**

1. Interactive SQL、または SQL 文を実行できる別のアプリケーションからデータベースに接続します。
2. ALTER DATABASE 文を実行します。

   たとえば、次の文は、JDK レベルを変更しないでデータベースをアップグレードします。

   ```
   ALTER DATABASE UPGRADE
   ```

   詳細については、「ALTER DATABASE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
3. アップグレードしたデータベースを使用する前に、データベースを停止し、トランザクション・ログを圧縮します。

バージョン 10.0.0 のデータベースの再構築

データベースのファイル・フォーマットをアップグレードするには、データベースをアンロードしてから再ロードする必要があります。

警告
大規模なデータベースのアンロードや再ロードには時間がかかり、大容量のディスク領域が必要になることがあります。この処理は、アンロードされたデータと新しいデータベース・ファイルを保持するため、データベースの約 2 倍のディスク領域を要する場合があります。

ユーティリティを使用して、SQL Remote レプリケーションに使用するデータベース、または Mobile Link インストール環境のリモート・データベースであるデータベースのファイル・フォーマットをアップグレードする場合は、-ar オプションまたは -an オプションを使用します。このオプションによって、新しいデータベースのトランザクション・ログのオフセットが、古いデータベースと同じになるように設定されます。

注意
データベースは、再構築の前にバックアップすることをおすすめします。

♦ データベース・ファイル・フォーマットをアップグレードするには、次の手順に従います (Sybase Central の場合)。

1. ソフトウェアをアップグレードする場合の一般的な対応策を行います。「アップグレードを行う前の重要な注意事項」 353 ページを参照してください。
2. Sybase Central を起動します。

   [スタート] – [プログラム] – [SQL Anywhere 10] – [Sybase Central] を選択します。
3. アップグレードするデータベースがあるバージョン 10 のデータベース・サーバを起動します。
4. [ツール] メニューから [SQL Anywhere 10] – [データベースのアンロード] を選択します。

   [データベース・アンロード] ウィザードが表示されます。
5. このウィザードの最初のページの説明を読んで、[次へ] をクリックします。
6. 接続先のデータベースのアンロードを選択します。[次へ] をクリックします。
7. すでに実行されているデータベースをアンロードするよう選択します。[次へ] をクリックします。
8. データベースの新しいファイル名を指定します。[次へ] をクリックします。
9. 新しいデータベースのページ・サイズを指定できます。デフォルトのページ・サイズは 4096 バイトです。必要に応じてデータベース・ファイルを暗号化できます。暗号化を行うと、データベースを起動するたびに、暗号化キーが必要になります。

   データベース・ファイルの暗号化の詳細については、「データベースの暗号化」 『SQL Anywhere サーバ - データベース管理』を参照してください。
10. [構造とデータをアンロード] を選択します。必要に応じて、他のオプションも選択します。[次へ] をクリックします。

11. [すべてのデータベース・オブジェクトをアンロード] を選択します。[次へ] をクリックします。
12. アンロード／再ロードが完了したときに新しいデータベースに接続するかどうかを指定します。
13. [完了] をクリックすると、処理が開始します。新しいデータベースを調べて、再構築が正常に完了したことを確認します。

    [データベース・アンロード] ウィザードの使用の詳細については、「[データベース・アンロード] ウィザードの使用」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

♦ データベース・ファイル・フォーマットをアップグレードするには、次の手順に従います (コマンド・ラインの場合)。

1. ソフトウェアをアップグレードする場合の一般的な対応策を行います。「アップグレードを行う前の重要な注意事項」 353 ページを参照してください。
2. アップグレードするデータベースに排他的にアクセスできること、またシステム・パスで、バージョン 10 のユーティリティがその他のユーティリティより前に指定されていることを確認します。「ユーティリティの使用」 352 ページを参照してください。
3. -ar オプションを指定してアンロード・ユーティリティ (dbunload) を実行して、新しいデータベースを作成します。

   ```
   dbunload -c "connection-string" -an new-db-file
   ```

   connection-string で指定するデータベース・ユーザは、アンロードするデータベースに DBA 権限で接続する必要があります。

   このコマンドによって、既存のデータベースが、アップグレードしたデータベースに置き換わります。-ar オプションを使用するには、パーソナル・サーバに接続するか、アンロード・ユーティリティ (dbunload) と同じコンピュータ上にあるネットワーク・サーバに接続します。

   アンロード・ユーティリティ (dbunload) のその他のオプションについては、「アンロード・ユーティリティ (dbunload)」 『SQL Anywhere サーバ - データベース管理』を参照してください。
4. 再ロードしたデータベースを使用する前に、データベースを停止し、トランザクション・ログを圧縮します。

   アンロードと再ロードのときに (大文字と小文字を「区別する」から「区別しない」に変更するなど) データベースの特性を変更する場合、手順はもう少し複雑になります。詳細については、「データベースの再構築」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

データベース・ミラーリング・システムでの SQL Anywhere ソフトウェアとデータベースのアップグレード

データベース・ミラーリングを使用しているときは、SQL Anywhere のメンテナンス・リリースまたは EBF を適用するため、またデータベース・ファイルをアップグレードするために追加の作業が必要です。

- メンテナンス・リリースの適用の詳細については、「データベース・ミラーリング・システムでの SQL Anywhere のメンテナンス・リリースのインストール」 358 ページを参照してください。
- EBF の適用の詳細については、「データベース・ミラーリング・システムでの SQL Anywhere の EBF の適用」 358 ページを参照してください。
- データベース・ファイルのアップグレードまたは再構築の詳細については、「データベース・ミラーリング・システムでのデータベースのアップグレード」 359 ページを参照してください。

データベース・ミラーリング・システムでの SQL Anywhere のメンテナンス・リリースのインストール

データベース・ミラーリング・システム内のすべてのサーバで、SQL Anywhere の同じメンテナンス・リリースを使用している必要があります。次の手順で SQL Anywhere のメンテナンス・リリースを適用した場合、データベースを使用できないのは手順 3 と 4 の実行中だけです。

♦ SQL Anywhere のメンテナンス・リリースをデータベース・ミラーリング・システムに適用するには、次の手順に従います。

1. dbstop コマンドを実行してミラー・サーバを停止します。
2. ミラー・サーバに SQL Anywhere の新バージョンをインストールします。
3. サーバごとに dbstop コマンドを実行して、プライマリ・サーバと監視サーバを停止します。
4. プライマリ・サーバに SQL Anywhere の新バージョンをインストールします。
5. プライマリ・サーバとミラー・サーバを再起動します。
6. 監視サーバタにソフトウェアの新バージョンをインストールします。
7. 監視サーバを再起動します。

データベース・ミラーリング・システムでの SQL Anywhere の EBF の適用

EBF をインストールするには、ミラーリング・システム内のデータベース・サーバ (プライマリ・サーバ、ミラー・サーバ、監視サーバ) ごとに次の操作を行う必要があります。

1. dbstop コマンドを実行してデータベース・サーバを停止します。
2. EBF をインストールします。
3. サーバを再起動します。

ダウン時間は、プライマリ・サーバ停止時のフェールオーバ中にのみ生じます。

参照

- 「ミラーリング・システムのデータベース・サーバの停止」『SQL Anywhere サーバ - データベース管理』
- 「プライマリ・サーバのフェールオーバの起動」『SQL Anywhere サーバ - データベース管理』

データベース・ミラーリング・システムでのデータベースのアップグレード

データベース・ミラーリング・システム内のデータベースは、2 通りの方法でアップグレードまたは再構築できます。最初の方法のほうが簡単ですが、2 つ目の方法よりもデータベースのダウン時間が長くなります。

♦ データベース・ミラーリング・システム内のデータベースをアップグレードまたは再構築するには、次の手順に従います。

1. ミラー・サーバを停止します。
2. プライマリ・サーバを停止します。
3. プライマリ・サーバ上のコピーを使用して、データベースをアップグレードまたは再構築します。「バージョン 10.0.0 のデータベースのアップグレード」 354 ページまたは「バージョン 10.0.0 のデータベースの再構築」 355 ページを参照してください。
4. アップグレードまたは再構築したデータベースとトランザクション・ログをミラー・サーバにコピーします。
5. プライマリ・サーバを再起動します。
6. ミラー・サーバを再起動します。

> **注意**
> 名前を変更したトランザクション・ログ・ファイルがある場合は移動してください。これらのファイルは、新しいデータベースと互換性がありません。ミラーリングを開始するには、初期のトランザクション・ログ・ファイルが両方のサーバに必要です。トランザクション・ログ・ファイルは、データベースに対して dbping コマンドを実行することで作成できます。

♦ データベース・ミラーリング・システム内のデータベースのアップグレードまたは再構築中のダウン時間を最小限にするには、次の手順に従います。

1. データベースをバックアップし、トランザクション・ログの名前を変更します。
2. dbtran ユーティリティを実行してデータベースの現在のトランザクション・ログ・ファイルの開始オフセットと終了オフセットを表示します。

   dbtran を使用してトランザクション・ログのオフセットを取得する方法については、「同期やレプリケーションに関連するデータベースの再構築」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

手順 2 ー 6 の間に発生したトランザクションを、アップグレードまたは再構築したデータベースに適用できるように、トランザクション・ログのオフセットを保存して再設定する必要があります。

3. データベースのバックアップ・コピーを別のコンピュータにアップグレードまたは再構築します。「バージョン 10.0.0 のデータベースのアップグレード」 354 ページまたは「バージョン 10.0.0 のデータベースの再構築」 355 ページを参照してください。

4. dblog ユーティリティを使用してログのオフセット情報を、手順 2 で取得した設定に再設定します。次に例を示します。

   ```
   dblog -x 0 -z 137829 database-name.db
   ```

   dblog を使用してトランザクション・ログのオフセットを再設定する方法については、「同期やレプリケーションに関連するデータベースの再構築」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

5. 次の各手順はオプションです。

   a. プライマリ・サーバのトランザクション・ログをバックアップし、名前を変更します。

   b. dbeng10 -a を使用して、手順 5.a で取得したトランザクション・ログを、再構築したデータベースに適用します。

6. プライマリ・サーバとミラー・サーバの両方を停止します。

7. プライマリ・データベースに対するトランザクション・ログの最新のコピーを保存します。

8. アップグレードまたは再構築したデータベースをプライマリ・サーバとミラー・サーバにコピーします。

9. 手順 7 のトランザクション・ログをプライマリ・サーバとミラー・サーバの両方にコピーします。

10. プライマリ・サーバを起動します。

11. ミラー・サーバを起動します。

バージョン 9 以前のデータベースをバージョン 10.0.0 用に再構築

この項では、データベースをアンロードして、新しいバージョン 10 データベースに再ロードする方法について説明します。

Windows CE データベースのアップグレードについては、「Windows CE のデータベースの再構築」 『SQL Anywhere サーバ - データベース管理』を参照してください。

Mac OS X のデータベースの再構築
Mac OS X で SQL Anywhere 9.0.2 以前のデータベースをアンロードするには、9.0.2 以前のソフトウェアを使用します。SQL Anywhere 10 ソフトウェアを使用して Mac OS X データベースをアンロードするには、別のプラットフォームでデータベースをアンロードする必要があります。再ロードは、Mac OS X でバージョン 10 のソフトウェアを使用して行うことができます。

警告
大規模なデータベースのアンロードや再ロードには時間がかかり、大容量のディスク領域が必要になることがあります。この処理では、アンロードされたデータと新しいデータベース・ファイルを保持するため、データベースの約 2 倍のディスク領域にアクセスする必要があります。

アップグレードの制限

10.0.0 のツールを使用してバージョン 9.0.2 のデータベースを再構築するときには、次の制限があるので注意してください。

- 旧バージョンの全データベース・サーバのデータベースを切断して、コンピュータで実行している旧バージョンのデータベース・サーバを停止する必要があります。また、コンピュータで実行しているバージョン 10 のデータベース・サーバも停止する必要があります。dbunload によってこれらの状態が検出された場合は、エラーが発生し、再構築に失敗します。
- 再構築前のデータベースの dbunload 接続文字列に、ENG、START、または LINKS 接続パラメータを含めないでください (-c オプションで指定)。これらの接続パラメータを指定すると、無視され、警告が表示されます。Sybase Central の [接続] ダイアログの [サーバ名] フィールドと [開始行] フィールドには、値を入力しないでください。
- 既存のデータベースが置かれているコンピュータで dbunload を実行する必要があります (dbunload が、共有メモリを使用してデータベースに接続できる状態である*必要があります*)。
- 再構築を実行するコンピュータで、dbunload_support_engine という名前のデータベース・サーバを実行することはできません。
- NetWare を使用している場合は、Windows または UNIX コンピュータでデータベースを再構築してから、NetWare コンピュータで実行しているデータベース・サーバ上の新しいバージョン 10.0.0 のデータベースに接続する必要があります。

特別な注意事項

- **パスワードの大文字と小文字の区別**　新しく作成された SQL Anywhere 10 データベースでは、データベースでの設定にかかわらず、すべてのパスワードは大文字と小文字が区別されます。新しいデータベースのデフォルトの DBA パスワードは、**sql** です。

  既存のデータベースを再構築する場合、SQL Anywhere での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。

  - パスワードを大文字と小文字を区別しないデータベースに最初に入力した場合、そのパスワードの大文字と小文字は区別されません。
  - パスワードを大文字と小文字を区別するデータベースに最初に入力した場合、大文字のパスワードと、大文字と小文字が混在したパスワードでは、大文字と小文字が区別されます。ただし、パスワードをすべて小文字で入力した場合、パスワードの大文字と小文字は*区別されません*。
  - 既存のパスワードと新しいパスワードの両方に加えられた変更は、大文字と小文字が区別されます。

- **ページ・サイズ**　SQL Anywhere 10 データベースのデフォルトのページ・サイズが、2048 バイトから 4096 バイトに変更されました。バージョン 10 でサポートされるページ・サイズは、2048、4096、8192、16384、32768 バイトです。既存のデータベースで、サポートされていないページ・サイズが使用されている場合は、新しいデータベースのページ・サイズはデフォルトで 4096 バイトに設定されます。dbinit の -p オプションを使用して、別のページ・サイズを指定できます。「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **照合**　再構築されたデータベース用に新しい照合や別の照合を指定していない場合は、既存のデータベースの照合がアンロードされ、再構築されたデータベースで再利用されます。

  カスタム照合を使用してデータベースを再構築する場合に 1 ステップで再構築すると、その照合は保存されます。データベースをアンロードしてから、作成したデータベースにスキーマとデータをロードする場合は、提供されるいずれかの照合を使用する必要があります。

Sybase Central からのバージョン 9 以前のデータベースの再構築

[データベース・アンロード] ウィザードを使用して、既存のデータベースを再構築できます。このウィザードでは、再ロード・ファイルとデータ・ファイルにアンロードするか、新しいデータベースにアンロードおよび再ロードするか、既存のデータベースにアンロードおよび再ロードするかのいずれかの操作を実行できます。再構築の前に、データベースをバックアップすることを強くおすすめします。

Sybase Central でのアップグレードに関する注意事項

- データベース・ファイルは、SQL Anywhere 10 がインストールされているコンピュータと同じコンピュータに保存されている必要があります。
- データベースからはテーブルのサブセットはアンロードできません。dbunload ユーティリティを使用してアンロードしてください。
- [データベース・アンロード] ウィザードで、データベース・ファイルがすでに実行されていると認識された場合は、アンロード・プロセスの前にデータベースが停止します。

♦ データベース・ファイル・フォーマットをアップグレードするには、次の手順に従います (Sybase Central の場合)。

1. ソフトウェアをアップグレードする場合の一般的な対応策を行います。「アップグレードを行う前の重要な注意事項」 353 ページを参照してください。
2. 断片化されているドライブでは、データベースのパフォーマンスが低下するので、可能な場合は、新しいデータベースを保存するドライブの断片化を解除してください。

   断片化を解除しない場合の手順については、「断片化を解除しないバージョン 9 以前のデータベースの再構築」 365 ページを参照してください。
3. データベースをバックアップします。次に例を示します。

   ```
   dbbackup -c "DBF=mydb.db;UID=DBA;PWD=sql" old-db-backup-dir
   ```


> **注意**
> データベースのバックアップには、正しいバージョンの dbbackup を使用してください。「ユーティリティの使用」 352 ページを参照してください。

4. アンロードと再ロードを行うデータベースに対して、排他アクセスを持っていることを確認してください。他のユーザは接続できません。
5. Sybase Central を起動します。

   [スタート] – [プログラム] – [SQL Anywhere 10] – [Sybase Central] を選択します。

   SQL Anywhere タスクをリストする Sybase Central のダイアログが表示されます。ダイアログが表示されない場合は、[ツール] メニューを使用して、次に示す代替方法を実行してください。
6. [SQL Anywhere 10 用にバージョン 9 以前のデータベースを準備] をクリックします。または、[ツール] メニューから、[SQL Anywhere 10] – [データベースのアンロード] を選択します。

   [データベース・アンロード] ウィザードが表示されます。
7. このウィザードの概要ページを読んで、[次へ] をクリックします。
8. [実行中ではないデータベースをアンロードする] を選択し、データベースの接続情報を入力します。[次へ] をクリックします。
9. [新しいデータベースへのアンロードと再ロード] を選択します。[次へ] をクリックします。
10. データベースの新しいファイル名を指定します。[次へ] をクリックします。

    新しいデータベースのページ・サイズを指定できます。バージョン 10 では、デフォルトのページ・サイズ (推奨値) は 4096 バイトです。

    必要に応じてデータベース・ファイルを暗号化できます。暗号化を行うと、データベースを起動するたびに、暗号化キーが必要になります。「データベースの暗号化」 『SQL Anywhere サーバ - データベース管理』を参照してください。
11. [構造とデータをアンロード] を選択します。[次へ] をクリックします。
12. アンロード／再ロードが完了したときに新しいデータベースに接続するかどうかを指定します。
13. [完了] をクリックすると、処理が開始します。新しいデータベースを調べて、アップグレードが正常に完了したことを確認してください。

アップロード・ユーティリティを使用したバージョン 9 以前のデータベースの再構築

アンロード・ユーティリティ (dbunload) の -an または -ar オプションを使用して、既存のデータベースを再構築できます。

- *-an オプションを使用すると新規データベースが作成されるので、このオプションを使用することをおすすめします。*

- -ar オプションを使用すると、既存のデータベースが新しいバージョン 10 のデータベースに置き換わります。

再構築の前に、データベースをバックアップすることをおすすめします。

注意
データベースのページ・サイズには、2048、4096、8192、16384、32768 バイトのいずれかを指定できます。デフォルトは元のデータベースのページ・サイズです。

♦ データベース・ファイル・フォーマットをアップグレードするには、次の手順に従います (コマンド・ラインの場合)。

1. ソフトウェアをアップグレードする場合の一般的な対応策を行います。「アップグレードを行う前の重要な注意事項」 353 ページを参照してください。
2. 断片化されているドライブでは、データベースのパフォーマンスが低下するので、可能な場合は、新しいデータベースを保存するドライブの断片化を解除してください。

   断片化を解除しない場合の手順については、「断片化を解除しないバージョン 9 以前のデータベースの再構築」 365 ページを参照してください。
3. データベースをバックアップします。次に例を示します。

   ```
   dbbackup -c "DBF=mydb.db;UID=DBA;PWD=sql" old-db-backup-dir
   ```

   注意
   データベースのバックアップには、正しいバージョンの dbbackup を使用してください。「ユーティリティの使用」 352 ページを参照してください。
4. アンロードと再ロードを行うデータベースに対して、排他アクセスを持っていることを確認してください。他のユーザは接続できません。
5. システム・パスで、他のユーティリティより前にバージョン 10 のユーティリティが置かれていることを確認してください。「ユーティリティの使用」 352 ページを参照してください。
6. バージョン 10 の dbunload ユーティリティは、旧バージョンのデータベース・サーバで実行されているデータベースに対して使用できないので、すべての SQL Anywhere と Adaptive Server Anywhere のデータベース・サーバを停止します。次に例を示します。

   ```
   dbstop -c "DBF=mydb.db;UID=DBA;PWD=sql"
   ```
7. -an または -ar オプションを指定してアンロード・ユーティリティ (dbunload) を実行して、新しいデータベースを作成します。

   ```
   dbunload -c "connection-string" -an database-filename
   ```

   次に例を示します。

   ```
   dbunload -c "DBF=mydb.db;UID=DBA;PWD=sql" -o dbunload_log_mydb.txt -an mydb10.db
   ```


connection-string で指定するデータベース・ユーザは、アンロードするデータベースに DBA 権限で接続する必要があります。このコマンドを使用すると、新規データベースが作成されます (-an を指定した場合)。-ar オプションを指定すると、既存のデータベースはアップグレード後のデータベースに置き換わります。-ar オプションを使用するには、パーソナル・サーバに接続するか、アンロード・ユーティリティ (dbunload) と同じコンピュータ上にあるネットワーク・サーバに接続します。

アンロード・ユーティリティ (dbunload) のその他のオプションについては、「アンロード・ユーティリティ (dbunload)」 『SQL Anywhere サーバ - データベース管理』を参照してください。

8. 再ロードしたデータベースを使用する前に、データベースを停止し、トランザクション・ログをバックアップします。

断片化を解除しないバージョン 9 以前のデータベースの再構築

新しいデータベースが保存されるドライブの断片化を解除しない場合は、代わりに次の手順に従ってデータベースを再構築します。

♦ **断片化を解除しないでデータベースを再構築するには、次の手順に従います。**

1. バージョン 10 の dbunload ユーティリティは、旧バージョンのデータベース・サーバで実行されているデータベースに対して使用できないので、すべての SQL Anywhere と Adaptive Server Anywhere のデータベース・サーバを停止します。次に例を示します。

   ```
   dbstop -c "DBF=mydb.db;UID=DBA;PWD=sql"
   ```

2. システム・パスで、他のユーティリティより前にバージョン 10 のユーティリティが置かれていることを確認してください。「ユーティリティの使用」 352 ページを参照してください。

3. データベースをバックアップします。次に例を示します。

   ```
   dbbackup -c "DBF=mydb.db;UID=DBA;PWD=sql" old-db-backup-dir
   ```

4. dbunload を使用して、*reload.sql* ファイルを作成します。次に例を示します。

   ```
   dbunload -c "connection-string" directory-name
   ```

5. 初期化ユーティリティ (dbinit)、または Sybase Central の [データベース作成] ウィザードを使用して、新しいデータベース・ファイルを作成します。次に例を示します。

   ```
   dbinit new.db
   ```

6. Interactive SQL から新しいデータベースに接続します。

   ```
   dbisql -c "DBF=new.db;UID=DBA;pwd=sql"
   ```

7. 次の文を実行して、データをデータベースにロードするときに使用できるディスク領域をデータベースに追加します。データベース・ファイルのサイズ分の充分なディスク領域を追加してください。再ロードのパフォーマンスを改善できるように、連続したこのディスク領域を追加する必要があります。次に例を示します。

   ```
   ALTER DBSPACE system
   ADD 200MB
   ```


8. *reload.sql* ファイルを Interactive SQL からデータベースに適用します。

```
dbisql -c "DBF=new.db;UID=DBA;pwd=sql" reload.sql
```

既知の問題

dbunload または [データベース・アンロード] ウィザードを実行したときに再構築処理に失敗した場合は、次の手順に従って失敗の原因を診断できます。

♦ 再構築の失敗を診断するには、次の手順に従います。

1. 既存のデータベースで dbunload -n を実行します。

```
dbunload -c "connection-string" -n directory-name
```

2. 新しい、バージョン 10 の空のデータベースを作成します。

```
dbinit test.db
```

3. *reload.sql* ファイルを空のデータベースに適用します。

```
dbisql -c "DBF=test.db;UID=DBA;pwd=sql" reload.sql
```

4. *reload.sql* ファイルを新しいデータベースに適用したときに受信したメッセージに基づいて、元のデータベースまたは *reload.sql* ファイルに変更を加えます。

次の表に、再構築の失敗につながる既知の問題とその解決策を示します。

既知の問題	解決策
テーブル名のプレフィクスに所有者名が付いている場合は、プロシージャまたはトリガ内の DECLARE LOCAL TEMPORARY TABLE 文により、構文エラーが発生します。	所有者名を削除します。
CREATE TRIGGER 文に、トリガが定義されたテーブルの所有者名が付いておらず、ユーザが *reload.sql* ファイルを実行して参照するときに、テーブルが所有者名で修飾されている必要がある場合、Table '*table-name*' not found エラーが発生し、CREATE TRIGGER 文は失敗します。	テーブル名のプレフィクスに所有者名を付けます。

オブジェクト名 (テーブル、カラム、変数、パラメータ名など) が SQL Anywhere の以後のバージョンで導入された予約語 (NCHAR など) に対応している場合は、再ロードに失敗します。次に例を示します。
`CREATE PROCEDURE p( )`
`BEGIN`
`DECLARE NCHAR INT;`
`SET NCHAR = 1;`
`END`
	予約語へのすべての参照を、別の名前を使用するように変更します。変数名の場合は、名前の競合を回避する @ を名前のプレフィクスとして使用するのが、一般的な規則です。
	予約語の完全なリストについては、「予約語」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

既知の問題	解決策
データベースをバージョン 9 以前の dbunload でアンロードする場合、*reload.sql* ファイルに ml_add_property システム・プロシージャの呼び出しが含まれることがありますが、新しいバージョン 10 データベースにはこのプロシージャは存在しません。	バージョン 10 の dbunload ユーティリティを使用してデータベースをアンロードしてください。 適切なバージョンのデータベース・ユーティリティを使用していることを確認する方法については、「ユーティリティの使用」 352 ページを参照してください。
バージョン 9 以前の dbunload を使用してデータベースをアンロードする場合、Transact-SQL 外部ジョイン (*= または =* で指定) を使用するビューを再ロードすると、それらのビューが適切に作成されないことがあります。	次のセクション行を再ロード・スクリプトに追加します。 `SET TEMPORARY OPTION tsql_outer_joins='on'` Transact-SQL 外部ジョインを使用しているすべてのビューを、後で書き直してください。

Mobile Link のアップグレード

既存のソフトウェアとの互換性

- 新しい Mobile Link クライアントは、バージョン 10.0.0 より前の Mobile Link サーバとは互換性がありません。
- バージョン 10 の Mobile Link サーバは、バージョン 8、9、10 のクライアントで使用できます。バージョン 10 のクライアントを使用する場合は、-x オプションを使用して Mobile Link サーバを起動します。バージョン 8 または 9 のクライアントを使用する場合は、-xo オプションを使用して Mobile Link サーバを起動します。これより前のクライアントをサポートする必要がある場合は、そのバージョンのクライアントをサポートしている古いバージョンの Mobile Link サーバを使用する必要があります。
- 記述されている動作変更が既存のアプリケーションに影響を与えないことを確認してください。影響する場合は、問題が起こらないように既存のアプリケーションを更新してください。SQL Anywhere 10 - 変更点とアップグレード 1 ページを参照してください。

アップグレードの順序

既存の Mobile Link インストール環境をアップグレードするには、次の順序でコンポーネントをアップグレードします。

1. Mobile Link サーバを停止します。
2. 統合データベースをアップグレードします。

   「統合データベースのアップグレード」 369 ページを参照してください。
3. Mobile Link サーバをアップグレードします。

   「Mobile Link サーバのアップグレード」 373 ページを参照してください。
4. Mobile Link サーバを起動します。
5. Mobile Link クライアントをアップグレードします。

   SQL Anywhere リモート・データベースの詳細については、「SQL Anywhere Mobile Link クライアントのアップグレード」 373 ページを参照してください。Ultra Light アプリケーションの詳細については、「バージョン 10.0.0 以前の Ultra Light アプリケーション・コードのバージョン 10.0.1 への移植」 383 ページを参照してください。

アップグレードを行う前に、影響がありそうな動作の変更を確認し、一般的なアップグレード前の対応策を実行してください。

詳細については、次の項を参照してください。

- 「動作の変更と廃止される機能」 112 ページ
- 「アップグレードを行う前の重要な注意事項」 353 ページ

統合データベースのアップグレード

新しい Mobile Link サーバと既存の統合データベースを使用できるようにするには、新しいシステム・オブジェクトをインストールするアップグレード・スクリプトを実行する必要があります。アップグレード・スクリプトは、現在インストールされている Mobile Link システム・テーブルの所有者が実行する必要があります。

注意

- 10.0.0 より前のバージョンで作成された authenticate_user_hashed スクリプトを使用する場合は、お使いの RDBMS でバイナリに相当する型を使用して、BINARY(20) ではなく BINARY(32) を受け入れるようにスクリプトを変更する必要があります。

SQL Anywhere バージョン 10.0.0 のアップグレード

「バージョン 10.0.0 のデータベースのアップグレード」 350 ページを参照してください。

バージョン 10.0.0 より前の SQL Anywhere のアップグレード

- 10.0.0 より前のバージョンでは、Mobile Link システム・テーブルの所有者は DBO でした。SQL Anywhere データベースの設定スクリプトを実行するには、Mobile Link システム・テーブルの所有者として統合データベースにログインする必要があります。テーブルを変更できるパーミッションを持つユーザーとしてこのスクリプトを実行しても十分ではありません。アップグレード・スクリプトを実行するには、SETUSER SQL 文を使用して DBO を同一化する方法もあります。次に例を示します。

  ```
  SETUSER "dbo";
  ```

  Sybase Central で統合データベースをアップグレードするには、GRANT CONNECT 文を使用して DBO のパスワードを作成してから、DBO として接続する必要があります。次に例を示します。

  ```
  GRANT CONNECT TO "dbo" IDENTIFIED BY 'password';
  ```

  この場合、アップグレード後に、GRANT CONNECT を使用して DBO のパスワードを削除する必要があります。次に例を示します。

  ```
  GRANT CONNECT TO "dbo";
  ```

- SQL Anywhere 統合データベースは設定済みで、同期はまだ一度も実行していない場合は、アップグレード・スクリプトではなく、設定スクリプトを実行する必要があります。これは、SQL Anywhere 統合データベースだけに適用されます。

♦ 統合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 (バージョン 10.0.0 より前の SQL Anywhere の場合)。

1. バージョン 10.0.0 より前の SQL Anywhere の統合データベースをアップグレードする場合は、先にデータベースをバージョン 10 にアップグレードする必要があります。

   a. データベース・サーバを停止します。

   b. データベースをバージョン 10 にアップグレードします。

手順については、「バージョン 9 以前のデータベースのアップグレード」 350 ページを参照してください。

c. DBA としてログインした状態で、データベース・サーバを起動します。

注意 : アップグレードするには DBA でログインする必要があります。

2. バージョン 6.0.x からアップグレードする場合は、SQL Anywhere インストール環境の *MobiLink¥setup* サブディレクトリにある Mobile Link 設定スクリプトを実行します。それ以降のバージョンからアップグレードする場合は、設定スクリプトは実行しないでください。

設定スクリプトの詳細については、「Mobile Link 統合データベース」 『Mobile Link - サーバ管理』を参照してください。

3. アップグレードするデータベースのバージョンに対応したアップグレード・スクリプトを実行します。

アップグレード・スクリプトは、*upgrade_asa.sql* です。アップグレード・スクリプトは、SQL Anywhere インストール環境の *MobiLink¥upgrade¥version* にインストールされています。*version* は、アップグレード対象の SQL Anywhere バージョンを示します。

アップグレード・スクリプトを実行するには、DBO ユーザを同一化する必要があります。この処理には SETUSER SQL 文を使用します。

たとえば、SQL Anywhere バージョン 9.0.2 統合データベースをアップグレードする場合は、Interactive SQL でデータベースに接続し、次のコマンドを実行します。

```
SETUSER "dbo";
READ 'c:¥Program Files¥SQL Anywhere 10¥MobiLink¥upgrade¥9.0.2¥upgrade_asa.sql'
```

4. DBO のパスワードを削除します。次に例を示します。

```
GRANT CONNECT TO "dbo"
```

5. DBA 以外のユーザとして Mobile Link サーバを実行している場合は、新しい Mobile Link システム・オブジェクトの EXECUTE パーミッションをこのユーザに付与する必要があります。新しいシステム・オブジェクトは、アップグレード対象のバージョンによって異なります。次のコードで、すべての Mobile Link システム・オブジェクトへの必要なパーミッションが付与されます。このコードを実行する前に、ユーザ名 my_user を、Mobile Link サーバを実行しているユーザの名前に変更する必要があります。

```
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_column to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_connection_script to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_database to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_device to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_device_address to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_listening to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_property to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_clients to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_delivery to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_global_props to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_notifications to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_repository to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_repository_props to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_repository_staging to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_status_history to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_status_staging to my_user;
```



```
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_script to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_script_version to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_scripts_modified to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_sis_sync_state to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_subscription to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_table to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_table_script to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_user to my_user;
GRANT EXECUTE ON dbo.ml_qa_get_agent_network_property to my_user;
GRANT EXECUTE ON dbo.ml_qa_get_agent_object_property to my_user;
GRANT EXECUTE ON dbo.ml_qa_get_agent_property to my_user;
GRANT EXECUTE ON dbo.ml_qa_get_message_property to my_user;
GRANT EXECUTE ON dbo.ml_add_column to my_user;
GRANT EXECUTE ON dbo.ml_add_connection_script to my_user;
GRANT EXECUTE ON dbo.ml_add_dnet_connection_script to my_user;
GRANT EXECUTE ON dbo.ml_add_dnet_table_script to my_user;
GRANT EXECUTE ON dbo.ml_add_java_connection_script to my_user;
GRANT EXECUTE ON dbo.ml_add_java_table_script to my_user;
GRANT EXECUTE ON dbo.ml_add_lang_conn_script_chk to my_user;
GRANT EXECUTE ON dbo.ml_add_lang_connection_script to my_user;
GRANT EXECUTE ON dbo.ml_add_lang_table_script to my_user;
GRANT EXECUTE ON dbo.ml_add_lang_table_script_chk to my_user;
GRANT EXECUTE ON dbo.ml_add_property to my_user;
GRANT EXECUTE ON dbo.ml_add_table_script to my_user;
GRANT EXECUTE ON dbo.ml_add_user to my_user;
GRANT EXECUTE ON dbo.ml_delete_device to my_user;
GRANT EXECUTE ON dbo.ml_delete_device_address to my_user;
GRANT EXECUTE ON dbo.ml_delete_listening to my_user;
GRANT EXECUTE ON dbo.ml_delete_sync_state to my_user;
GRANT EXECUTE ON dbo.ml_delete_sync_state_before to my_user;
GRANT EXECUTE ON dbo.ml_delete_user to my_user;
GRANT EXECUTE ON dbo.ml_qa_add_delivery to my_user;
GRANT EXECUTE ON dbo.ml_qa_add_message to my_user;
GRANT EXECUTE ON dbo.ml_qa_handle_error to my_user;
GRANT EXECUTE ON dbo.ml_qa_stage_status_from_client to my_user;
GRANT EXECUTE ON dbo.ml_qa_staged_status_for_client to my_user;
GRANT EXECUTE ON dbo.ml_qa_upsert_global_prop to my_user;
GRANT EXECUTE ON dbo.ml_reset_sync_state to my_user;
GRANT EXECUTE ON dbo.ml_set_device to my_user;
GRANT EXECUTE ON dbo.ml_set_device_address to my_user;
GRANT EXECUTE ON dbo.ml_set_listening to my_user;
GRANT EXECUTE ON dbo.ml_set_sis_sync_state to my_user;
GRANT EXECUTE ON dbo.ml_upload_update_device_address to my_user;
GRANT EXECUTE ON dbo.ml_upload_update_listening to my_user;
```

Adaptive Server Enterprise、Oracle、または Microsoft SQL Server のアップグレード

Mobile Link サーバが 10.0.0 より前のバージョンの場合は、Adaptive Server Enterprise、Oracle、または Microsoft SQL Server の統合データベースのみアップグレードする必要があります。

♦ 統合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 (Adaptive Server Enterprise、Oracle、または Microsoft SQL Server)。

1. バージョン 6.0.x からアップグレードする場合は、SQL Anywhere インストール環境の *MobiLink ¥setup* サブディレクトリにある Mobile Link 設定スクリプトを実行します。それ以降のバージョンからアップグレードする場合は、設定スクリプトは実行しないでください。

   設定スクリプトの詳細については、「Mobile Link 統合データベース」 『Mobile Link - サーバ管理』を参照してください。

2. Adaptive Server Enterprise データベースの場合は、"select into" 権限を設定する必要があります。Sybase Interactive SQL で次のコマンドを実行します。

```
USE MASTER
go
sp_dboption your-database-name, "SELECT INTO", true
go
USE your-database-name
go
checkpoint
go
```

3. アップグレードするデータベースのバージョンに対応したアップグレード・スクリプトを実行します。

   アップグレード・スクリプトは *upgrade_XXX.sql* という名前です。*XXX* は、統合データベースの RDBMS を示します。アップグレード・スクリプトは、*MobiLink¥upgrade¥version* の SQL Anywhere インストール環境にインストールされています。*version* は、アップグレード対象の SQL Anywhere バージョンを示します。

   次に例を示します。

```
READ 'c:¥Program Files¥SQL Anywhere 10¥MobiLink¥upgrade¥902¥upgrade_asa.sql'
```

DB2 のアップグレード

Mobile Link サーバが 10.0.0 より前のバージョンの場合は、DB2 の統合データベースのみアップグレードする必要があります。

♦ **統合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 (DB2 の場合)。**

1. *MobiLink¥setup¥SyncDB2Long_1000.class* ファイルを DB2 サーバ・コンピュータの *SQLLIB¥FUNCTION* ディレクトリにコピーします。この場合は通常、DB2 のインスタンスを再起動する必要があります。詳細については、使用している DB2 のマニュアルを参照してください。

2. Mobile Link バージョン 6 からアップグレードする場合は、設定 SQL スクリプト *MobiLink¥setup¥syncdb2long.sql* を実行して、Mobile Link のシステム・テーブルとストアド・プロシージャを作成します。

   DB2 設定スクリプトを実行する方法については、「IBM DB2 UDB 統合データベース」『Mobile Link - サーバ管理』を参照してください。

3. DB2 アップグレード・スクリプトのロケーションを検索します。

   アップグレード・スクリプトは *upgrade_db2tolong.sql* という名前で、SQL Anywhere インストール環境の *MobiLink/upgrade/version* サブディレクトリにあります。*version* ディレクトリは、アップグレード対象の Mobile Link のバージョンを示します。

4. *upgrade_db2tolong.sql* をコピーし、このコピーを変更します。スクリプトの先頭にある CONNECT 文を変更して、接続するインスタンスで動作できるようにします。コピーした SQL スクリプトを統合データベースに適用します。

Mobile Link サーバのアップグレード

バージョン 10.0.0 からバージョン 10.0.1 にアップグレードする必要はありません。

バージョン 10 の Mobile Link サーバを使用する前に、影響がありそうな動作の変更を確認してください。SQL Anywhere 10 - 変更点とアップグレード 1 ページを参照してください。

Mobile Link バージョン 10 サーバでサポートされるのは、バージョン 8 および 9 の SQL Anywhere クライアントと Ultra Light クライアントだけです。これより前のクライアントをサポートする必要がある場合は、そのバージョンのクライアントをサポートしている古いバージョンの Mobile Link サーバを使用する必要があります。

SQL Anywhere Mobile Link クライアントのアップグレード

運用環境では、統合データベースと Mobile Link サーバの両方をアップグレードしてから、SQL Anywhere リモート・データベースをアップグレードします。

注意 : バージョン 10.0.0 で、Adaptive Server Anywhere は SQL Anywhere に名前が変更されました。

検討すべきアップグレードは複数あります。

- ソフトウェアのアップグレード
- リモート・データベース自体のアップグレード
- アプリケーション全体のアップグレード

警告
Mobie Link 同期に関連するデータベースのアップグレードは、同期が正常に完了した直後に行う必要があります。また、データベースを検証およびバックアップする必要があります。

ソフトウェアのアップグレード

dbmlsync Mobile Link クライアントと SQL Anywhere データベース・サーバは同時にアップグレードすることをおすすめします。リモート・データベースをアップグレードしてから、新しい dbmlsync ユーティリティを実行してください。

Mobile Link バージョン 10 クライアントでは、同期を実行するために Mobile Link バージョン 10 同期サーバが必要です。バージョン 10 の Mobile Link クライアントは、バージョン 10 より前の Mobile Link 同期サーバとは同期しません。

Mobile Link のアップグレードについては、「Mobile Link のアップグレード」 368 ページを参照してください。

リモート・データベースのアップグレード

Mobile Link SQL Anywhere リモート・データベースは、SQL Anywhere データベースのアップグレードと同じ方法でアップグレードできます。手順については、「SQL Anywhere のアップグレード」 350 ページを参照してください。

スキーマの変更やデータベースでのその他の重要な変更などがある場合は、手動でアンロードと再ロードを行う必要があります。

♦ SQL Anywhere リモート・データベースを手動でアンロードおよび再ロードするには、次の手順に従います。

1. 同期が正常に完了してから、リモート・データベースを検証し、バックアップします。
2. dbtran ユーティリティを実行してデータベースのトランザクション・ログの開始オフセットと終了オフセットを表示します。終了オフセットをメモしてください。

   「ログ変換ユーティリティ (dbtran)」『SQL Anywhere サーバ - データベース管理』を参照してください。
3. トランザクション・ログの名前を変更します。これにより、アンロード中にトランザクション・ログが修正されるのを回避できます。名前変更したログ・ファイルを、オフライン・ディレクトリなどの安全なロケーションに移動します。
4. データベースをアンロードします。

   「バージョン 9 以前のデータベースをバージョン 10.0.0 用に再構築」 360 ページを参照してください。
5. 新しいデータベースを初期化します。

   「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバ - データベース管理』を参照してください。
6. データを新しいデータベースに再ロードします。

   「バージョン 9 以前のデータベースをバージョン 10.0.0 用に再構築」 360 ページを参照してください。
7. 新しいデータベースを停止します。
8. 新しいデータベースのトランザクション・ログを消去します。
9. 次のオプションを使用して、新しいデータベースで dblog を実行します。

   - -z を使用して、手順 2 でメモした終了オフセットを指定します。
   - -x を使用して、相対オフセットをゼロに設定します。

   次に例を示します。

   ```
   dblog -x 0 -z 137829 database-name.db
   ```

   「トランザクション・ログ・ユーティリティ (dblog)」『SQL Anywhere サーバ - データベース管理』を参照してください。
10. dbmlsync を起動し、手順 3 で移動した元のログ・ファイルの場所を指定します。

    「dbmlsync 構文」『Mobile Link - クライアント管理』を参照してください。
11. 古いログ・ファイルが不要な場合は、データベース・オプション delete_old_logs を設定します。

「delete_old_logs オプション [Mobile Link クライアント] [SQL Remote] [Replication Agent]」『SQL Anywhere サーバ - データベース管理』を参照してください。

アプリケーションのアップグレード

Mobile Link アプリケーションの新しいバージョンを配備する場合、同期スクリプトには新しいバージョンの名前を使用することをおすすめします。たとえば、既存のアプリケーションで **v1** というスクリプトのバージョンを使用している場合は、アップグレードしたアプリケーションでは **v2** という名前のスクリプト・バージョンを使用します。この 2 つのスクリプト・バージョンは同時に使用できます。これにより、リモート・データベースを一度にではなく、段階的にアップグレードすることが容易になります。

バージョン 9.0.0 以降では、Mobile Link サーバ の -zd オプションは削除されています。配備環境で -zd オプションを使用している場合、アップグレードを行うには、最後のダウンロード・タイムスタンプが最初のパラメータとして採用されるようダウンロード・スクリプトを変更します。

QAnywhere のアップグレード

QAnywhere アプリケーションをアップグレードすると、統合データベース、アプリケーション、クライアント・メッセージ・ストアをアップグレードできます。

統合データベースをアップグレードする方法については、「統合データベースのアップグレード」 369 ページを参照してください。

アプリケーションをアップグレードする場合は、このリリースの新しい機能と変更された動作を確認してください。

SQL Anywhere 10 - 変更点とアップグレード 1 ページを参照してください。

♦ **QAnywhere メッセージ・ストアをアップグレードするには、次の手順に従ってください。**

1. QAnywhere ファイルを配備します。

   「QAnywhere アプリケーションの配備」 『Mobile Link - サーバ管理』を参照してください。

2. 次の手順に従って、メッセージ・ストアをアップグレードします。

   -su オプション または -sur オプションを指定して、QAnywhere Agent を実行します。次の項を参照してください。

   - 「-su オプション」 『QAnywhere』
   - 「-sur オプション」 『QAnywhere』

Ultra Light のアップグレード

以前のバージョンの Ultra Light をアップグレードするには、データベースとアプリケーション・コードに必要なアップグレード・パスを考慮する必要があります。

既存のソフトウェアとの互換性

- Ultra Light 10.0.1 ランタイムと Ultra Light 10.0.1 エンジンは、10.0.0 より前のバージョンの Ultra Light で作成したデータベース・ファイルおよびアプリケーション・コードでは使用できません。
- Ultra Light 10.0.1 では、バージョン 10.0.0 以外の Ultra Light のクライアント・アプリケーションからの接続はサポートされていません。
- 現在のバージョンの Sybase Central には、以前のバージョンのデータベースとクライアント・アプリケーションを管理する次の機能が用意されています。
  - バージョン 10 のデータベースは完全に管理できますが、以前のバージョンのデータベースは管理できません。
  - バージョン 5、6、7、8、または 9 のデータベースには、データベース・ファイル・フォーマットをアップグレードするために、接続のみできます。
  - [C++ API 移行] ウィザードを使用して、バージョン 7、8、9 の C/C++ アプリケーションのソース・ファイルをアップグレードできます。

Palm OS の初期バージョン
Palm デバイスの初期バージョン (バージョン 4.X など) には RAM が 200 KB ほどしかありません。この制限事項により、Ultra Light 10.0.1 の動的メモリ要件に起因する問題が発生することがあります。Ultra Light 10.0.1 用に以前のバージョンの Palm を最適化する方法については、「Ultra Light のプラットフォーム別の最適化方法」 『Ultra Light - データベース管理とリファレンス』を参照してください。.

Ultra Light 10.0.1 のユーティリティの使用

SQL Anywhere の複数のバージョンが同じコンピュータにインストールされている場合は、バージョン 10.0.1 の Ultra Light ユーティリティを使用するときに、システム・パスに注意してください。「ユーティリティの使用」 352 ページを参照してください。.

Ultra Light データベースのアップグレード

以前のバージョンの Ultra Light 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。

- データを同期します。
- すべてのアプリケーションと管理ツールを切断します。

- データベースをデスクトップ・コンピュータにコピーします。

> **注意**
> データベースをアップグレードしない場合は、Ultra Light 10.0.1 の管理ツールを使用して以前のバージョンのデータベースに接続することはできません。

データベースのアップグレードに関する特別な注意事項

- Ultra Light スキーマは、個別の *.usm* ファイルではなく、データベースの一部になりました。したがって、データベース接続を確立できないときにアプリケーションで新しいデータベースを作成できなくなりました。アプリケーションでは、初期データベースを配備するか、新しい CreateDatabase 機能を使用してプログラムでデータベースを作成する必要があります。
- ファイル・フォーマットは、Ultra Light バージョン 10 の時点で統合されました。つまり、ほとんどのプラットフォームでデータベースを共有できるようになったので、ユニコード文字は不要になりました。

  選択した照合に含まれない文字が必要な場合は、UTF-8 を使用してデータベースをコード化する必要があります。「Ultra Light での文字セットのエンコードに関するプラットフォーム要件」『Ultra Light - データベース管理とリファレンス』と「Ultra Light utf8_encoding プロパティ」『Ultra Light - データベース管理とリファレンス』を参照してください。

  > **Windows CE とデスクトップ・データベース**
  > これらのプラットフォームのデータベースをアップグレードするときに、ユニコード文字が不要な場合は、データベースを UTF-8 コード化しないでください。UTF-8 コード化により、不必要にデータベースのサイズが増加します。

- バージョン 10 では、データベースで大文字と小文字を区別するかは関係なく、すべてのデータベースのパスワードで大文字と小文字を区別します。このため、データベースをアップグレードしたときに、ユーザ ID、パスワード、信用されたルート証明書は保持されないことがあります。新しい Ultra Light データベースに、前のユーザ ID、パスワード、信用されたルート証明書を追加してください。新しいデータベースのデフォルトの DBA パスワードは、**sql** です。

Ultra Light データベースのアップグレード・パス

Ultra Light ではデータベースには複数の作成方法があるので、アップグレード処理もその作成方法に応じて異なります。次の表に、アップグレードの対象ごとにどの方法を使用するかを示します。

アップグレード対象	使用するツール
♦ スキーマ・ファイル (*.usm*) ♦ データベース・ファイル (*.udb*). ♦ Palm OS データベース・レコード (*.pdb*)	[データベース・アップグレード] ウィザード、または古いデータベースのアンロード・ユーティリティ (ulunloadold) とデータベース・ロード・ユーティリティ (ulload)
SQL Anywhere リファレンス・データベースをソースとする Ultra Light データベース[1]	[データベース抽出] ウィザードまたはデータベース初期化ユーティリティ (ulinit)

この項で説明したツールを使用したアップグレード手順ついては、「Ultra Light のデータベース・アップグレード・ツール」 379 ページを参照してください。

Ultra Light のデータベース・アップグレード・ツール

既存の Ultra Light データベースまたはスキーマをアップグレードするには、[データベース・アップグレード] ウィザードまたは古いデータベースのアンロード・ユーティリティ (ulunloadold) を使用します。

- 説明に従ってプロパティやオプションを選択していきたい場合は、ウィザードを選択します。
- 次のいずれかの要件が該当する場合は、ユーティリティを選択します。
  - 指定したテーブルのみを新しいデータベースにアップグレードする場合
  - バッチ関連の処理を実装する場合

♦ 既存の Ultra Light データベースをバージョン 10.0.1 にアップグレードするには、次の手順に従います (Sybase Central の場合)。

1. ソフトウェアをアップグレードする場合の一般的な対応策を行います。「アップグレードを行う前の重要な注意事項」 353 ページを参照してください。
2. [スタート] – [プログラム] – [SQL Anywhere 10] – [Sybase Central] を選択して、Sybase Central を起動します。
3. [ツール] – [Ultra Light 10] – [データベースのアップグレード] を選択して、データベースをアップグレードします。

   [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。次の項目を指定してから、次に進んでください。

   - ソースとして、データベースまたはスキーマ・ファイルを選択します。
   - アップグレードしたデータベースの出力先を次のいずれかから選択します。

1 最初に、SQL Anywhere データベースが更新済みであることを確認してください。「SQL Anywhere のアップグレード」 350 ページを参照してください。

- **新しい Ultra Light 10.0.1 データベース**　このオプションを選択すると、データベースが作成され、接続されます。
- **既存の Ultra Light 10.0.1 データベース**　このオプションを選択すると、SQL Anywhere リファレンス・データベースの設定を使用して、データベース・オプションまたは照合を変更できます。既存の Ultra Light データベースと内部スキーマ内のデータに適した文字セットおよび照合を選択してください。

4. 次のいずれかのオプションを選択して、アップグレード・ソースを選択します。
   - **古いデータベース**　Ultra Light データベース (**.udb* または **.pdb*) を参照します。
   - **古いスキーマ・ファイル**　Ultra Light スキーマ・ファイル (**.usm*) を参照します。
5. 選択したファイルに接続して、[次へ] をクリックします。
6. 出力先を選択します。
   - **新しいデータベース**　新しいデータベースを作成して、必要なデータベース・プロパティを設定する必要があります。ウィザードの指示に従います。
   - **接続済みのデータベースを使用する**　接続したデータベースは、表示されたリストから選択できます。
   - **接続されていない既存のデータベースを使用する**　[データベース] をクリックして [接続] ダイアログを開き、既存の Ultra Light 10.0.1 データベースに接続します。
7. ウィザードの指示に従い、出力先に関連するその他の項目を選択します。以前のリリースの Ultra Light に信用されるルート証明書が含まれている場合は、新しい Ultra Light データベースに追加してください。
8. [完了] をクリックして、データベースをアップグレードします。
9. 以前のバージョンの Ultra Light のユーザが、新しく作成したデータベースに表示されてない場合は、そのユーザを新しいデータベースに追加します。「Ultra Light のユーザの操作」『Ultra Light - データベース管理とリファレンス』を参照してください。

♦ 既存の Ultra Light データベースをバージョン 10.0.1 にアップグレードするには、次の手順に従います (コマンド・ラインの場合)。

1. ソフトウェアをアップグレードする場合の一般的な対応策を行います。「アップグレードを行う前の重要な注意事項」 353 ページを参照してください。
2. システム・パスで、古い Ultra Light のユーティリティより前にバージョン 10.0.1 の Ultra Light のユーティリティが置かれていることを確認してください。「ユーティリティの使用」 352 ページを参照してください。
3. コマンド・プロンプトを開き、古いデータベースのアンロード・ユーティリティ (ulunloadold) で次の構文を使用して XML 中間ファイルを作成します。

   ulunloadold -c "*connection-string*" [*options*] *xml-file*

次の操作を実行済みかどうかを確認します。

- ulunloadold ユーティリティで作成する XML ファイルの名前の指定
- 古い Ultra Light データベース (**.udb* または **.pdb*) をアップグレードするか、古い Ultra Light スキーマ・ファイル (**.usm*) をアップグレードするかによって、*connection-string* 内で DBF または schema_file パラメータを使用しているか

それ以外のすべてのオプションは任意です。

このユーティリティの完全なリファレンスについては、「Ultra Light 古いデータベースのアンロード・ユーティリティ (ulunloadold)」『Ultra Light - データベース管理とリファレンス』を参照してください。

4. XML のデータベースへのロード・ユーティリティ (ulload) を実行して、XML を既存または新しい Ultra Light データベースに再ロードします。

   XML を新しいデバイスにロードする場合は、**-c** *connection-string* オプションがデータベースの接続パラメータ (Ultra Light ユーザの認証に必要な UID や PWD など) を設定します。

   設定する **-o** [*extended-options*] は、データベースの特性やプロパティを変更しているかどうか (大文字と小文字を区別するデータベースを大文字小文字を区別しないデータベースに変更するなど) によって異なります。

   完全なリファレンスについては、「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」『Ultra Light - データベース管理とリファレンス』を参照してください。

   たとえば、中間の XML ファイル *dbschema.xml* を使用して、*dbschema8.usm* という Ultra Light 8.x のスキーマ・ファイルを、*db.udb* という Ultra Light バージョン 10.0.1 の既存のデータベースにアップグレードするには、次の 2 つのコマンドが必要です。

   ```
   ulunloadold -c schema_file=dbschema8.usm dbschema.xml

   ulload -c DBF=db.udb dbschema.xml
   ```

初期化／抽出ツール

[データベース抽出] ウィザードまたはデータベース初期化ユーティリティ (ulinit) を使用して、バージョン 10.0.1 の SQL Anywhere データベースから Ultra Light データベースを抽出できます。

Ultra Light での使用を考慮したリファレンス・データベースのプロパティの設定
Ultra Light データベースは、SQL Anywhere リファレンス・データベースと同じプロパティ設定で生成されます。これらのオプションをリファレンス・データベースで設定することによって、Ultra Light データベースの動作も制御することになります。

- 説明に従ってプロパティやオプションを選択していきたい場合は、ウィザードを選択します。
- 次のいずれかの要件が該当する場合は、ユーティリティを選択します。
  - 指定したテーブルのみを新しいデータベースにアップグレードする場合
  - バッチ関連の処理を実装する場合

♦ SQL Anywhere リファレンスから Ultra Light データベースを初期化または抽出するには、次の手順に従います (Sybase Central の場合)。

1. ソフトウェアをアップグレードする場合の一般的な対応策を行います。「アップグレードを行う前の重要な注意事項」 353 ページを参照してください。
2. 既存の SQL Anywhere データベースがアップグレード済みで、Ultra Light での使用用途を考慮したうえでそのデータベースの準備が完了していることを確認します。パブリケーションを更新する必要がある場合は、Ultra Light データベースを再作成する前に更新してください。

   SQL Anywhere のアップグレード手順については、「ユーティリティの使用」 352 ページを参照してください。Ultra Light で使用する SQL Anywhere の準備方法については、「SQL Anywhere リファレンス・データベースからの Ultra Light データベースの作成」 『Ultra Light - データベース管理とリファレンス』を参照してください。
3. [スタート] – [プログラム] – [SQL Anywhere 10] – [Sybase Central] を選択して、Sybase Central を起動します。
4. [ツール] – [Ultra Light 10] – [(ulinit) データベースの抽出] を選択して、SQL Anywhere データベースの Ultra Light バージョンを抽出します。

   [データベースの抽出] 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。

♦ SQL Anywhere リファレンス・データベースから Ultra Light データベースを初期化または抽出するには、次の手順に従います (コマンド・ラインの場合)。

1. ソフトウェアをアップグレードする場合の一般的な対応策を行います。「アップグレードを行う前の重要な注意事項」 353 ページを参照してください。
2. システム・パスで、古い Ultra Light のユーティリティより前にバージョン 10.0.1 の Ultra Light のユーティリティが置かれていることを確認してください。「ユーティリティの使用」 352 ページを参照してください。
3. 既存の SQL Anywhere データベースがアップグレード済みで、Ultra Light での使用用途を考慮したうえでそのデータベースの準備が完了していることを確認します。パブリケーションを更新する必要がある場合は、Ultra Light データベースを再作成する前に更新してください。

   SQL Anywhere のアップグレード手順については、「ユーティリティの使用」 352 ページを参照してください。Ultra Light で使用する SQL Anywhere の準備方法については、「SQL Anywhere リファレンス・データベースからの Ultra Light データベースの作成」 『Ultra Light - データベース管理とリファレンス』を参照してください。
4. コマンド・プロンプトを開き、データベースの初期化ユーティリティ (ulinit) で次の構文を使用して Ultra Light データベースを抽出します。


```
ulinit -a "SAconnection-string" -c "ULconnection-string"
-n pubname [ options ]
```

次の操作を実行済みかどうかを確認します。

- アップグレード済みの SQL Anywhere リファレンス・データベースと、このコマンドで作成する新しい Ultra Light データベースの接続文字列を指定済みかどうか
- Ultra Light データベースで必要なテーブルが含まれるパブリケーションを指定済みかどうかすべてのテーブルを抽出する場合は、**-n*** を使用します。

それ以外のすべてのオプションは任意です。

注意
ここでは新しいデータベースを作成しているので、**UID** や **PWD** などのパラメータは、認証のための初期ユーザ ID およびパスワードの作成に使用されます。この場合、SQL Anywhere データベースは参照されません。ただし、その他の SQL Anywhere リファレンス・データベースのデフォルト・プロパティを上書きするには、**-o** [*extended-options*] を使用してください。完全なリストについては、「サポートされている拡張オプション」 『Ultra Light - データベース管理とリファレンス』を参照してください。

完全なリファレンスについては、「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 『Ultra Light - データベース管理とリファレンス』を参照してください。設定可能な Ultra Light データベース・プロパティの詳細については、「Ultra Light データベース設定のリファレンス」 『Ultra Light - データベース管理とリファレンス』を参照してください。

バージョン 10.0.0 以前の Ultra Light アプリケーション・コードのバージョン 10.0.1 への移植

以前の Ultra Light アプリケーションは、新しいバージョン 10.0.1 の API で再構築する必要があります。Ultra Light 10.0.0 のリリース以来、これらの API は大幅に強化されました。必要に応じてコードの変更をしてから、アプリケーションを再構築します。

「Ultra Light の新機能」 168 ページを詳しく調べて、使用する API に加えられた変更を確認してから、アプリケーションを移植します。

アプリケーションのアップグレードに関する特別な注意事項

- *接続コードを更新する必要があります。*.ほとんどの API の場合、接続パラメータ・コントロールを使用するようにします。ただし、Ultra Light for AppForge でだけは新しいコントロールを使用する必要があります。たとえば、バージョン 9.x の Ultra Light for MobileVB プロジェクトで ULConnectionParms オブジェクトを使用する ULDatabaseManager *WithParms メソッドはサポートされなくなりました。代わりに、ULConnectionParms.ToString() メソッドを使用するようにこのコードを書き換える必要があります。

  接続パラメータ・コントロールは、一連の接続パラメータを簡単にアセンブルできます。Ultra Light のランタイムは代わりに、アセンブルされたパラメータを接続文字列に変換します。接続オブジェクトは、Ultra Light for AppForge API 以外のすべての API で、引き続き使用できま

す。ただし、接続パラメータ・コントロールの方が、接続文字列のエラーをより適切に診断できます。API の詳細については、次の項を参照してください。

- Ultra Light for C/C++：「データベースへの接続」『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL：「データベースへの接続」『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：「Ultra Light データベースへの接続」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「データベースへの接続」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「Ultra Light データベースへの接続」『Ultra Light - M-Business Anywhere プログラミング』

- Ultra Light スキーマは、個別の *.usm* ファイルとしてではなく、テーブルとしてデータベースに統合されました。つまり、このファイルを使用してデバイス上のデータベースを作成することはできません。代わりに、新しいデータベース作成関数／メソッドが追加されました。ただし、このメソッドを使用すると、アプリケーションのサイズが増加します。アプリケーションのサイズの増加を防ぐには、管理ツールを使用してデスクトップ上でデータベースを作成してから、後でアプリケーションに配備します。「デスクトップでの Ultra Light の作成」『Ultra Light - データベース管理とリファレンス』を参照してください。

- このバージョンの Ultra Light では、常に認証が有効になっており、最大 4 つのユーザ ID とパスワードをサポートできます。ただし、データベースの認証を維持しない場合は、ユーザ ID およびパスワードは作成または指定してないでください。Ultra Light では、ユーザ ID およびパスワードが指定されていない場合は、デフォルト **UID=DBA** および **PWD=sql** が使用されます。「ユーザ ID とパスワードの組み合わせの解釈」『Ultra Light - データベース管理とリファレンス』を参照してください。

- バージョン 10.0 の Ultra Light for AppForge コンポーネントによって、Ultra Light for MobileVB の以前のバージョンが置き換えられました。同じマシン上で複数のバージョンのコンポーネントを使用することはできません。使用した場合は、AppForge は VB プロジェクトをコンパイルできず、次のエラーを生成します。

  "Error importing ulmvbctl9.dll. Unable to load"

  したがって、このコンポーネントの 9.x バージョンを登録解除する必要があります。詳細については、「Ultra Light アプリケーション・コードのアップグレード・パス」 385 ページの表を参照してください。

  また、ユーザのデバイス上にバージョン 9.x の Ultra Light for MobileVB クライアントがある場合は、それを削除してから、バージョン 10.0.1 の Ultra Light for AppForge クライアントをインストールして使用するようにします。

- Embedded SQL ファイルが複数ある場合は、SQL プロセッサ (sqlpp) を使用してそれぞれのファイルの前処理を実行し、C/C++ ソース・ファイルを作成する必要があります。ただし、リファレンス・データベースを使用する必要はなくなりました。Ultra Light データベースは、Embedded SQL を直接サポートするようになりました。

- ユニコード文字は、前のバージョンと同じように使用することはできません。代わりに、バージョン 10.0.1 の Ultra Light のデータベースでは、マルチバイト文字に UTF-8 コード化が使用

されています。その結果、非ユニコード・ランタイムで実行しているユニコードのデータベースを考慮する必要がなくなりました。

Ultra Light アプリケーション・コードのアップグレード・パス

Ultra Light では複数の開発 API および方法があるので、アップグレード処理もその方法に応じて異なります。次の表に、アップグレードの対象ごとにどの方法を使用するかを示します。

アップグレード対象	手順
ulgen で生成された C/C++ アプリケーション	1. [データベース抽出] ウィザードを使用するか、Ultra Light 初期化ユーティリティ (ulinit) を実行して、バージョン 10.0.1 の Ultra Light データベースを作成します。「初期化／抽出ツール」 381 ページを参照してください。[1] 2. [C++ API 移行] ウィザードを使用し、Ultra Light の SQL Anywhere 10 プロジェクトからテーブルおよび文を読み込んで、API を移行します。「Ultra Light アプリケーション・コードのアップグレード・ツール」 386 ページを参照してください。

Ultra Light for MobileVB バージョン 9.x	1. Ultra Light for MobileVB バージョン 9.0.x を登録解除します。コマンド・プロンプトで次のように入力します。
	`<ASA90-install>¥ultralite¥UltraliteForMobileVB¥win32¥ulafreg -u`
	2. Ultra Light for AppForge 10.0.1 コンポーネントを登録します。コマンド・プロンプトで次のように入力します。
	`<install-dir>¥win32¥ulafreg -r`
	Visual Basic によって、ULConnectionParms を使用する各サブプロジェクトごとにメッセージが表示されます。
	`Version 9.0 of ulmvbctl9.dll is not registered. The control will be updated to version 10.0.`
	3. 新しいバージョンを使用するには、[OK] をクリックします。

Embedded SQL アプリケーション	変更された点はごくわずかで、ツールは不要です。
Java アプリケーション	Ultra Light 10.0.0 のリリース以来、Java は Ultra Light に含まれなくなりました。サポートされている API を使用してアプリケーションを書き換える必要があります。

アップグレード対象	手順
Ultra Light コンポーネント	主なコードの書き換えには次のようなものがあります。 ♦ **スキーマの書き換え** このバージョンでは、スキーマはデータベースに統合されたので、すべてのコンポーネントが ULConnection オブジェクトの OpenWithCreate 関数を書き換え、すべてのスキーマのアップグレード・コードを削除する必要があります。代わりに、ULDatabaseManager.CreateDatabase を使用してデバイス上でデータベースを作成できます。ただし、新しいデータベースのプロパティの定義に必要なコードの量を減らすには、デスクトップでデータベースを作成して、完了したらデバイスに配備する必要があります。詳細については、「デバイス上での Ultra Light の作成」『Ultra Light - データベース管理とリファレンス』と「デスクトップでの Ultra Light の作成」『Ultra Light - データベース管理とリファレンス』を参照してください。 ♦ **接続の書き換え** 接続パラメータ・オブジェクトを使用したインタフェースが削除されました。これには、データベースを作成、開く、または削除した関数やメソッドも含まれます。代わりに、文字列インタフェースを使用して、接続パラメータを渡します。

[1] 最初に、SQL Anywhere データベースが更新済みであることを確認してください。「SQL Anywhere のアップグレード」 350 ページを参照してください。

Ultra Light アプリケーション・コードのアップグレード・ツール

ulgen で生成された C/C++ ソース・コードを移行できるのは、[C++ API 移行] ウィザードを使用した場合だけです。このウィザードでは、バージョン 10 の仕様に適合しなくなった Embedded SQL を識別できます。移行処理を完了できない場合は、変更した SQL 文を終了して、*.*uag* ファイルに保存できます。

初めてウィザードを使用する場合、テーブルおよび文のソースは SQL Anywhere リファレンス・データベースです。後で実行するときには、保存されている *.*uag* ファイルを代わりに使用できます。

♦ **Ultra Light C/C++ API を移行するには、次の手順に従います (Sybase Central の場合)。**

1. ソフトウェアをアップグレードする場合の一般的な対応策を行います。「アップグレードを行う前の重要な注意事項」 353 ページを参照してください。
2. 既存の SQL Anywhere データベースがアップグレード済みで、Ultra Light での使用用途を考慮したうえでそのデータベースの準備が完了していることを確認します。パブリケーションを更新する必要がある場合は、Ultra Light データベースを再作成する前に更新してください。

   SQL Anywhere のアップグレード手順については、「ユーティリティの使用」 352 ページを参照してください。Ultra Light で使用する SQL Anywhere の準備方法については、「SQL Anywhere リファレンス・データベースからの Ultra Light データベースの作成」『Ultra Light - データベース管理とリファレンス』を参照してください。

3. Ultra Light のデータベースがアップグレード済みであることを確認します。アップグレード済みでない場合は、SQL Anywhere リファレンス・データベースから Ultra Light データベースを抽出する必要があります。

   この Ultra Light データベースは、この抽出処理の検証データベースとして使用されます。

4. [スタート] − [プログラム] − [SQL Anywhere 10] − [Sybase Central] を選択して、Sybase Central を起動します。

5. [ツール] − [Ultra Light 10] − [C++ API 移行] を選択して、C/C++ API アプリケーションを移行します。

   [C++ API 移行] ウィザードが表示されます。

6. [SQL 文のソース] ページで、文およびテーブルを読み取るソースを選択します。

   - 初めてウィザードを実行している場合は、[SQL リファレンス・データベースに接続] を選択してから [データベース] をクリックし、SQL Anywhere のリファレンス・データベースの接続情報を設定します。
   - 初めての実行ではない場合は、[前のウィザード・セッションから出力を読み込む] を選択してから [参照] をクリックすることで、この目的のために作成した **.uag* ファイルに書き込まれた以前の変更を開くこともできます。

7. 選択したソースに応じてウィザードの手順に従い、すべての SQL 文を検証します。無効な文の場合、文の名前の横に赤い × が付きます。Ultra Light でサポートされる各文の完全なリファレンスについては、「Ultra Light SQL 文のリファレンス」『Ultra Light - データベース管理とリファレンス』を参照してください。

   無効な SQL 文を修正するには、次の手順に従ってください。

   a. 無効な文を選択します。

   b. 表示されたテキスト・ボックスに文を修正します。

   c. [すべての SQL 文を検証] をクリックします。

   検証された文は、リストの一番下に表示され、文の名前の横に緑色のチェックマークが付きます。いつでも [キャンセル] をクリックして、*.uag* ファイルへの変更を保存し、ウィザードを終了できます。

SQL Remote のアップグレード

既存の SQL Remote インストール環境をバージョン 6 以降からアップグレードする場合は、各データベース・サーバをその Message Agent (dbremote) より前、または同時にアップグレードします。Message Agent のアップグレードには特に決まった順序はありません。

バージョン 5 のユーザは、「バージョン 5 の SQL Remote インストール環境のアップグレード」 388 ページの指示に従ってください。

- **データベースのアップグレード** データベース・ファイルのフォーマットをアップグレードするには、データベースをアンロードしてから再ロードします。すべてのデータベースを同時にアップグレードする必要はありません。

  データベースをアンロードおよび再ロードする方法については、「SQL Anywhere のアップグレード」 350 ページを参照してください。

- **ソフトウェア・アップグレードは一度に 1 サイトずつ行う** 以前のバージョンの Message Agent は、バージョン 10 の Message Agent とメッセージを交換できます。

- **Message Agent とデータベース・サーバは別々にアップグレードできる** データベース・サーバは Message Agent よりも先にアップグレードできます。ただし、パフォーマンスの面からは、Message Agent をデータベース・サーバと同時にアップグレードすることをおすすめします。

- **Adaptive Server Enterprise 統合データベースのアップグレード** SQL Remote では、Adaptive Server Enterprise の統合データベースはサポートされなくなりました。Adaptive Server Enterprise データベースを同期するには、Mobile Link にアップグレードする必要があります。

  SQL Remote から Mobile Link への移行については、http://www.ianywhere.com/whitepapers/migrate_to_ml.html を参照してください。

バージョン 5 の SQL Remote インストール環境のアップグレード

SQL Remote インストール環境には、1 つの統合データベースと多数のリモート・データベースが含まれています。また、各サイトには、Message Agent があります。

それぞれのサイトでは、Message Agent がメッセージの送受信を行います。メッセージは SQL 文の形式で、データベース・サーバがその SQL 文の実行を処理します。

SQL Remote のアップグレード要件は、次のとおりです。

- **データベースのアップグレード** データベース・ファイルのフォーマットをアップグレードするには、データベースをアンロードしてから再ロードします。

  「SQL Anywhere のアップグレード」 350 ページを参照してください。

- **ソフトウェア・アップグレードは一度に 1 サイトずつ行う** バージョン 5 の Message Agent は、compression データベース・オプションが -1 (マイナス 1) の値に設定されていれば、バー

ジョン 10 の Message Agent とメッセージを交換できます。インストール環境全体のソフトウェアを同時にアップグレードする必要はありません。

「compression オプション [SQL Remote]」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **Message Agent とサーバは別々のアップグレードが可能** Message Agent は Embedded SQL アプリケーションです。このため、互換性ライブラリを使用すれば、データベース・サーバを Message Agent より先にアップグレードできます。ただし、パフォーマンスの面からは、Message Agent をデータベース・サーバと同時にアップグレードすることをおすすめします。

  Message Agent をデータベース・サーバより先にアップグレードすることはできません。新しいクライアント・アプリケーションはバージョン 5 サーバでは機能しないからです。

例

以下に、アップグレードの一例を示します。

1. 統合データベース・サーバと Message Agent をアップグレードします。すべてのメッセージがリモート・サイトのバージョン 5 ソフトウェアと互換性を持つように、compression データベース・オプションを -1 に設定してください。
2. リモート・データベース・サーバと Message Agent をアップグレードします。compression データベース・オプションを -1 以外の値に設定すれば、統合データベース・サーバに送信されるメッセージに対して圧縮機能とエンコード機能を利用できます。
3. すべてのリモート・データベース・サーバと Message Agent をアップグレードし終わったら、統合サイトの compression データベース・オプションを -1 以外の値に設定します。

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

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

つ

て

と

な

に

ね

の

は

ひ

ふ

へ

ほ

ま

れ

ろ

わ

