



SQL Anywhere® 11 変更点とアップグレード

2009 年 2 月

バージョン 11.0.1

版權と商標

Copyright © 2009 iAnywhere Solutions, Inc. Portions copyright © 2009 Sybase, Inc. All rights reserved.

iAnywhere との間に書面による合意がないかぎり、このマニュアルは現状のまま提供されるものであり、その使用または記載内容の誤りに対して一切の責任を負いません。

次の条件に従うかぎり、このマニュアルの全部または一部を使用、印刷、再生、配布することができます。1) マニュアルの全部または一部にかかわらず、すべてのコピーにこの情報またはマニュアル内のその他の版權と商標の表示を含めること。2) マニュアルに変更を加えないこと。3) iAnywhere 以外の人間がマニュアルの著者または情報源であるかのように示す行為をしないこと。

iAnywhere®、Sybase®、および <http://www.sybase.com/detail?id=1011207> に記載されているマークは、Sybase, Inc. または子会社の商標です。® は米国での登録商標を示します。

このマニュアルに記載されているその他の会社名と製品名は各社の商標である場合があります。

目次

はじめに	vii
SQL Anywhere のマニュアルについて	viii
バージョン 11.0.1 の新機能	1
SQL Anywhere の新機能	2
SQL Anywhere の動作の変更と廃止予定機能	9
Mobile Link の新機能	14
Mobile Link の動作の変更と廃止予定機能	16
QAnywhere の新機能	17
Ultra Light の新機能	18
Ultra Light の動作の変更と廃止予定機能	19
Ultra Light J の新機能	20
Ultra Light J の動作の変更と廃止予定機能	21
管理ツールの新機能	22
管理ツールの動作の変更と廃止予定機能	24
製品全体の新機能	25
マニュアルの強化	26
バージョン 11.0.0 の新機能	27
SQL Anywhere	29
Mobile Link	72
QAnywhere	78
SQL Remote	80
Ultra Light	81
Sybase Central と Interactive SQL	88
マニュアルの強化	98
製品全体の機能	99
バージョン 10.0.1 の新機能	101
SQL Anywhere	103

Mobile Link	115
QAnywhere	118
SQL Remote	120
Ultra Light	121
製品全体の機能	123
 バージョン 10.0.0 の新機能	 127
SQL Anywhere	129
Mobile Link	201
QAnywhere	225
SQL Remote	231
Ultra Light	233
Sybase Central と Interactive SQL	248
マニュアルの強化	254
製品全体の機能	255
 バージョン 9.0.2 の新機能	 259
バージョン 9.0.2 の新機能	260
バージョン 9.0.2 での動作の変更	277
 バージョン 9.0.1 の新機能	 285
バージョン 9.0.1 の新機能	286
バージョン 9.0.1 での動作の変更	301
 バージョン 9.0.0 の新機能	 305
バージョン 9.0.0 の新機能	306
バージョン 9.0 での動作の変更	325
 バージョン 8.0.2 の新機能	 333
バージョン 8.0.2 の新機能	334
バージョン 8.0.2 での動作の変更	344

バージョン 8.0.1 の新機能	349
バージョン 8.0.1 の新機能	350
バージョン 8.0.1 での動作の変更	356
 バージョン 8.0.0 の新機能	 359
バージョン 8 の新機能	360
バージョン 8 での動作の変更	382
 SQL Anywhere 11 へのアップグレード	 391
SQL Anywhere のアップグレード	392
Mobile Link のアップグレード	410
QAnywhere のアップグレード	418
Ultra Light のアップグレード	419
SQL Remote のアップグレード	429
 索引	 431

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 11 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 のバージョン 8 より前における新機能や動作の変更については、http://www.ianywhere.com/developer/product_manuals/sqlanywhere/1001/en/html/dbwnen10/dbwnen10.html を参照してください。

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 のマニュアルについて

SQL Anywhere の完全なマニュアルは 4 つの形式で提供されており、いずれも同じ情報が含まれています。

- **HTML ヘルプ** オンライン・ヘルプには、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。

Microsoft Windows オペレーティング・システムを使用している場合は、オンライン・ヘルプは HTML ヘルプ (CHM) 形式で提供されます。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 11] - [マニュアル] - [オンライン・マニュアル] を選択します。

管理ツールのヘルプ機能でも、同じオンライン・マニュアルが使用されます。

- **Eclipse** UNIX プラットフォームでは、完全なオンライン・ヘルプは Eclipse 形式で提供されます。マニュアルにアクセスするには、SQL Anywhere 11 インストール環境の *bin32* または *bin64* ディレクトリから *sadoc* を実行します。

- **DocCommentXchange** DocCommentXchange は、SQL Anywhere マニュアルにアクセスし、マニュアルについて議論するためのコミュニティです。

DocCommentXchange は次の目的に使用できます (現在のところ、日本語はサポートされておられません)。

- マニュアルを表示する
- マニュアルの項目について明確化するために、ユーザによって追加された内容を確認する
- すべてのユーザのために、今後のリリースでマニュアルを改善するための提案や修正を行う

<http://dcx.sybase.com> を参照してください。

- **PDF** SQL Anywhere の完全なマニュアル・セットは、Portable Document Format (PDF) 形式のファイルとして提供されます。内容を表示するには、PDF リードが必要で、Adobe Reader をダウンロードするには、<http://get.adobe.com/reader/> にアクセスしてください。

Microsoft Windows オペレーティング・システムで PDF 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 11] - [マニュアル] - [オンライン・マニュアル - PDF] を選択します。

UNIX オペレーティング・システムで PDF マニュアルにアクセスするには、Web ブラウザを使用して *install-dir/documentation/ja/pdf/index.html* を開きます。

マニュアル・セットに含まれる各マニュアルについて

SQL Anywhere のマニュアルは次の構成になっています。

- **『SQL Anywhere 11 - 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 11 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 11 - 変更点とアップグレード』 このマニュアルでは、SQL Anywhere 11 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 データベースを実行、管理、構成する方法について説明します。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、管理ユーティリティとオプションについて説明します。
- 『SQL Anywhere サーバ - プログラミング』 このマニュアルでは、C、C++、Java、PHP、Perl、Python、および Visual Basic や Visual C# などの .NET プログラミング言語を使用してデータベース・アプリケーションを構築、配備する方法について説明します。ADO.NET や ODBC などのさまざまなプログラミング・インタフェースについても説明します。
- 『SQL Anywhere サーバ - SQL リファレンス』 このマニュアルでは、システム・プロシージャとカタログ (システム・テーブルとビュー) に関する情報について説明します。また、SQL Anywhere での SQL 言語の実装 (探索条件、構文、データ型、関数) についても説明します。
- 『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。
- 『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 テクノロジーは、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 - 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。また、dbmlsync API についても説明します。dbmlsync API を使用すると、同期を C++ または .NET のクライアント・アプリケーションにシームレスに統合できます。
- 『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 - サーバ起動同期』 このマニュアルでは、Mobile Link サーバ起動同期について説明します。この機能により、Mobile Link サーバは同期を開始したり、リモート・デバイス上でアクションを実行することができます。
- 『QAnywhere』 このマニュアルでは、モバイル・クライアント、ワイヤレス・クライアント、デスクトップ・クライアント、およびラップトップ・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、QAnywhere について説明します。
- 『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。

- **『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 - 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・デバイス、モバイル・デバイス、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 Mobile、または Windows を搭載しているハンドヘルド・デバイス、モバイル・デバイス、または埋め込みデバイスの Web ベースのデータベース・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 - .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・デバイス、モバイル・デバイス、または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 J』** このマニュアルでは、Ultra Light J について説明します。Ultra Light J を使用すると、Java をサポートしている環境用のデータベース・アプリケーションを開発し、配備することができます。Ultra Light J は、BlackBerry スマートフォンと Java SE 環境をサポートしており、iAnywhere Ultra Light データベース製品がベースになっています。
- **『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示し、その診断情報を説明します。

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表記規則について説明します。

オペレーティング・システム

SQL Anywhere はさまざまなプラットフォームで稼働します。ほとんどの場合、すべてのプラットフォームで同じように動作しますが、いくつかの相違点や制限事項があります。このような相違点や制限事項は、一般に、基盤となっているオペレーティング・システム (Windows、UNIX など) に由来しており、使用しているプラットフォームの種類 (AIX、Windows Mobile など) またはバージョンに依存していることはほとんどありません。

オペレーティング・システムへの言及を簡素化するために、このマニュアルではサポートされているオペレーティング・システムを次のようにグループ分けして表記します。

- **Windows** Microsoft Windows ファミリを指しています。これには、主にサーバ、デスクトップ・コンピュータ、ラップトップ・コンピュータで使用される Windows Vista や Windows XP、およびモバイル・デバイスで使用される Windows Mobile が含まれます。
特に記述がないかぎり、マニュアル中に Windows という記述がある場合は、Windows Mobile を含むすべての Windows ベース・プラットフォームを指しています。

- **UNIX** 特に記述がないかぎり、マニュアル中に UNIX という記述がある場合は、Linux および Mac OS X を含むすべての UNIX ベース・プラットフォームを指しています。

ディレクトリとファイル名

ほとんどの場合、ディレクトリ名およびファイル名の参照形式はサポートされているすべてのプラットフォームで似通っており、それぞれの違いはごくわずかです。このような場合は、Windows の表記規則が使用されています。詳細がより複雑な場合は、マニュアルにすべての関連形式が記載されています。

ディレクトリ名とファイル名の表記を簡素化するために使用されている表記規則は次のとおりです。

- **大文字と小文字のディレクトリ名** Windows と UNIX では、ディレクトリ名およびファイル名には大文字と小文字が含まれている場合があります。ディレクトリやファイルが作成されると、ファイル・システムでは大文字と小文字の区別が維持されます。

Windows では、ディレクトリおよびファイルを参照するとき、大文字と小文字は**区別されません**。大文字と小文字を混ぜたディレクトリ名およびファイル名は一般的に使用されますが、参照するときはすべて小文字を使用するのが通常です。SQL Anywhere では、*Bin32* や *Documentation* などのディレクトリがインストールされます。

UNIX では、ディレクトリおよびファイルを参照するとき、大文字と小文字は**区別されます**。大文字と小文字を混ぜたディレクトリ名およびファイル名は一般的に使用されません。ほとんどの場合は、すべて小文字の名前が使用されます。SQL Anywhere では、*bin32* や *documentation* などのディレクトリがインストールされます。

このマニュアルでは、ディレクトリ名に Windows の形式を使用しています。ほとんどの場合、大文字と小文字が混ざったディレクトリ名をすべて小文字に変換すると、対応する UNIX 用のディレクトリ名になります。

- **各ディレクトリおよびファイル名を区切るスラッシュ** マニュアルでは、ディレクトリの区切り文字に円記号を使用しています。たとえば、PDF 形式のマニュアルは *install-dir¥Documentation¥ja¥pdf* にあります。これは Windows の形式です。

UNIX では、円記号をスラッシュに置き換えます。PDF マニュアルは *install-dir/documentation/ja/pdf* にあります。

- **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、*.exe* や *.bat* などの拡張子が付きます。UNIX では、実行ファイルの名前に拡張子は付きません。

たとえば、Windows でのネットワーク・データベース・サーバは *dbsrv11.exe* です。UNIX では *dbsrv11* です。

- **install-dir** インストール・プロセス中に、SQL Anywhere をインストールするロケーションを選択します。このロケーションを参照する環境変数 *SQLANY11* が作成されます。このマニュアルでは、そのロケーションを *install-dir* と表します。

たとえば、マニュアルではファイルを *install-dir¥readme.txt* のように参照します。これは、Windows では、*%SQLANY11%¥readme.txt* に対応します。UNIX では、*\$SQLANY11/readme.txt* または *\${SQLANY11}/readme.txt* に対応します。

install-dir のデフォルト・ロケーションの詳細については、「[SQLANY11 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・プロセス中に、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します。このロケーションを参照する環境変数 SQLANYSAMP11 が作成されます。このマニュアルではそのロケーションを *samples-dir* と表します。

Windows エクスプローラ・ウィンドウで *samples-dir* を開くには、[スタート] - [プログラム] - [SQL Anywhere 11] - [サンプル・アプリケーションとプロジェクト] を選択します。

samples-dir のデフォルト・ロケーションの詳細については、「[SQLANYSAMP11 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。

コマンド・プロンプトとコマンド・シェル構文

ほとんどのオペレーティング・システムには、コマンド・シェルまたはコマンド・プロンプトを使用してコマンドおよびパラメータを入力する方法が、1 つ以上あります。Windows のコマンド・プロンプトには、コマンド・プロンプト (DOS プロンプト) および 4NT があります。UNIX のコマンド・シェルには、Korn シェルおよび bash があります。各シェルには、単純コマンドからの拡張機能が含まれています。拡張機能は、特殊文字を指定することで起動されます。特殊文字および機能は、シェルによって異なります。これらの特殊文字を誤って使用すると、多くの場合、構文エラーや予期しない動作が発生します。

このマニュアルでは、一般的な形式のコマンド・ラインの例を示します。これらの例に、シェルにとって特別な意味を持つ文字が含まれている場合、その特定のシェル用にコマンドを変更することが必要な場合があります。このマニュアルではコマンドの変更について説明しませんが、通常、その文字を含むパラメータを引用符で囲むか、特殊文字の前にエスケープ文字を記述します。

次に、プラットフォームによって異なるコマンド・ライン構文の例を示します。

- **カッコと中カッコ** 一部のコマンド・ライン・オプションは、詳細な値を含むリストを指定できるパラメータを要求します。リストは通常、カッコまたは中カッコで囲まれています。このマニュアルでは、カッコを使用します。次に例を示します。

```
-x tcpip(host=127.0.0.1)
```

カッコによって構文エラーになる場合は、代わりに中カッコを使用します。

```
-x tcpip{host=127.0.0.1}
```

どちらの形式でも構文エラーになる場合は、シェルの要求に従ってパラメータ全体を引用符で囲む必要があります。

```
-x "tcpip(host=127.0.0.1)"
```

- **引用符** パラメータの値として引用符を指定する必要がある場合、その引用符はパラメータを囲むために使用される通常の引用符と競合する可能性があります。たとえば、値に二重引用符を含む暗号化キーを指定するには、キーを引用符で囲み、パラメータ内の引用符をエスケープします。

```
-ek "my ¥"secret¥" key"
```

多くのシェルでは、キーの値は my "secret" key のようになります。

- **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 のシェルでは、環境変数は構文 %ENVVAR% を使用して指定されます。UNIX のシェルでは、環境変数は構文 \$ENVVAR または \${ENVVAR} を使用して指定されます。

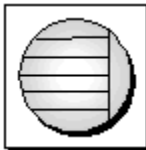
グラフィック・アイコン

このマニュアルでは、次のアイコンを使用します。

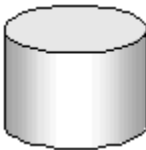
- クライアント・アプリケーション。



- SQL Anywhere などのデータベース・サーバ。



- データベース。ハイレベルの図では、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。



- プログラミング・インタフェース。

ドキュメンテーション・チームへのお問い合わせ

このヘルプに関するご意見、ご提案、フィードバックをお寄せください。

SQL Anywhere ドキュメンテーション・チームへのご意見やご提案は、弊社までご連絡ください。頂戴したご意見はマニュアルの向上に役立たせていただきます。ぜひとも、ご意見をお寄せください。

DocCommentXchange

DocCommentXchange を使用して、ヘルプ・トピックに関するご意見を直接お寄せいただくこともできます。DocCommentXchange (DCX) は、SQL Anywhere マニュアルにアクセスしたり、マニュアルについて議論するためのコミュニティです。DocCommentXchange は次の目的に使用できます (現在のところ、日本語はサポートされておりません)。

- マニュアルを表示する
- マニュアルの項目について明確化するために、ユーザによって追加された内容を確認する
- すべてのユーザのために、今後のリリースでマニュアルを改善するための提案や修正を行う

<http://dcx.sybase.com> を参照してください。

詳細情報の検索／テクニカル・サポートの依頼

詳しい情報やリソースについては、iAnywhere デベロッパー・コミュニティ (<http://www.iAnywhere.jp/developers/index.html>) 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。バージョンおよびビルド番号を調べるには、コマンド **dbeng11 -v** を実行します。

ニュースグループは、ニュース・サーバ forums.sybase.com にあります。

以下のニュースグループがあります。

- [ianywhere.public.japanese.general](http://groups.google.com/group/sql-anywhere-web-development)

Web 開発に関する問題については、<http://groups.google.com/group/sql-anywhere-web-development> を参照してください。

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也没有ありません。

iAnywhere のテクニカル・アドバイザー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。

バージョン 11.0.1 の新機能

目次

SQL Anywhere の新機能	2
SQL Anywhere の動作の変更と廃止予定機能	9
Mobile Link の新機能	14
Mobile Link の動作の変更と廃止予定機能	16
QAnywhere の新機能	17
Ultra Light の新機能	18
Ultra Light の動作の変更と廃止予定機能	19
Ultra Light J の新機能	20
Ultra Light J の動作の変更と廃止予定機能	21
管理ツールの新機能	22
管理ツールの動作の変更と廃止予定機能	24
製品全体の新機能	25
マニュアルの強化	26

廃止予定機能は変更される可能性があります

廃止予定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 の新機能

次に、バージョン 11.0.1 で導入された SQL Anywhere データベースとデータベース・サーバの追加機能を示します。

接続プロパティ

SQL Anywhere バージョン 11.0.1 で追加された接続プロパティは次のとおりです。

- Authenticated
- IsDebugger
- QueryBypassedCosted
- QueryBypassedHeuristic
- QueryBypassedOptimized
- QueryOpened
- QueryDescribedBypass
- QueryDescribedOptimizer
- StatementDescribes
- StatementPostAnnotates
- StatementPostAnnotatesSimple
- StatementPostAnnotatesSkipped
- WaitStartTime
- WaitType

これらのプロパティの説明については、「[接続プロパティ](#)」『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・プロパティ

SQL Anywhere バージョン 11.0.1 で追加されたデータベース・プロパティは次のとおりです。

- Authenticated
- Prepares
- QueryBypassedCosted
- QueryBypassedHeuristic
- QueryBypassedOptimized
- QueryOpened
- QueryDescribedBypass
- QueryDescribedOptimizer
- StatementDescribes
- StatementPostAnnotates
- StatementPostAnnotatesSimple
- StatementPostAnnotatesSkipped

これらのプロパティの説明については、「[データベース・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・サーバ・プロパティ

次に、SQL Anywhere バージョン 11.0.1 でのデータベース・サーバ・プロパティの強化を示します。

● ServerEdition

これらのプロパティの説明については、「データベース・サーバ・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・ユーティリティ

次に、SQL Anywhere バージョン 11.0.1 でのデータベース・ユーティリティの強化を示します。

● **サービス・ユーティリティ (dbsvc) の強化** Windows で、サービス・ユーティリティを使用して、Mobile Link リレー・サーバ (rshost)、Mobile Link リレー・サーバ Outbound Enabler (RSOE)、および SQL Anywhere ボリューム・シャドウ・コピー・サービス (dbvss11) のサービスを作成できるようになりました。「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) の強化** dbunload ユーティリティで次のオプションがサポートされるようになりました。

○ **-cm オプション** アンロードされるデータベースの dbinit コマンドまたは CREATE DATABASE 文を表示します。

○ **-i オプション** 再構築されたデータベースのオートインクリメント・コラムに、次に使用可能な値を保持します。

「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。

● **アップグレード・ユーティリティ (dbupgrad) の強化** データベース・ミラーリングに参加しているデータベースをアップグレードするために、アップグレード・ユーティリティを使用するとエラーになります。「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。

システム・プロシージャと関数

次に、SQL Anywhere バージョン 11.0.1 で追加されたシステム・プロシージャと関数の強化を示します。

● **sa_get_table_definition システム・プロシージャ** 新しい sa_get_table_definition システム・プロシージャは、指定されたテーブルおよびそのインデックス、外部キー、トリガ、付与されたパーミッションの作成に必要な SQL 文を含む LONG VARCHAR 文字列を返します。「sa_get_table_definition 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。

- **FIRST_VALUE 関数 [集合]** FIRST_VALUE 関数 [集合] に RESPECT NULLS 句が含まれるようになりました。「[FIRST_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。
- **LAST_VALUE 関数 [集合]** LAST_VALUE 関数 [集合] に RESPECT NULLS 句が含まれるようになりました。「[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。
- **sa_set_http_option システム・プロシージャ** AcceptCharset オプションにより、文字セットをより柔軟に選択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。

SQL 文

次に、SQL Anywhere バージョン 11.0.1 での SQL の強化を示します。

- **新しい DEFAULT VALUES 句、INSERT 文** INSERT 文に新しく追加された DEFAULT VALUES 句により、すべてのカラムにデフォルト値を挿入できます。「[INSERT 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。
- **CREATE ENCRYPTED DATABASE 文** この文は、既存のデータベースの暗号化されたコピー (すべてのトランザクション・ログ、ミラー・ログ、DB 領域を含む) を作成します。この文を使用して、テーブル暗号化が有効になっているデータベースのコピーも作成できます。「[CREATE ENCRYPTED DATABASE 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。

リカバリが必要なデータベースを暗号化したい場合 (テクニカル・サポートに送る場合など) は、今までどおり CREATE ENCRYPTED FILE 文を使用する必要があります。「[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 DECRYPTED DATABASE 文** この文は、既存のデータベースの復号化されたコピー (すべてのトランザクション・ログ、ミラー・ログ、DB 領域を含む) を作成します。「[CREATE DECRYPTED DATABASE 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。

リカバリが必要なデータベースを復号化したい場合 (テクニカル・サポートに送る場合など) は、今までどおり CREATE DECRYPTED FILE 文を使用する必要があります。「[CREATE DECRYPTED FILE 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。
- **ALTER DATABASE 文の強化** ミラーリング中のデータベース・サーバに ALTER DATABASE UPGRADE 文を実行すると、エラーになります。「[ALTER DATABASE 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。

- **関数、プロシージャ、トリガ、ビューの新規作成または同じ名前のオブジェクトの置換** 新しい OR REPLACE 句により、関数、プロシージャ、トリガ、ビューの新規作成や、同じ名前のオブジェクトが存在する場合は置換することができます。次の項を参照してください。
 - 「CREATE FUNCTION 文 [Web サービス]」 『SQL Anywhere サーバ - SQL リファレンス』
 - 「CREATE PROCEDURE 文 [Web サービス]」 『SQL Anywhere サーバ - SQL リファレンス』
 - 「CREATE TRIGGER 文」 『SQL Anywhere サーバ - SQL リファレンス』
 - 「CREATE VIEW 文」 『SQL Anywhere サーバ - SQL リファレンス』
- **存在しないデータベース・オブジェクトの削除を文で試みてもエラーを表示しない** 新しい IF EXISTS 句により、存在しないデータベース・オブジェクトの削除を DROP 文で試みてもエラーを返さないように指定できます。次の項を参照してください。
 - 「DROP EVENT 文」 『SQL Anywhere サーバ - SQL リファレンス』
 - 「DROP FUNCTION 文」 『SQL Anywhere サーバ - SQL リファレンス』
 - 「DROP MATERIALIZED VIEW 文」 『SQL Anywhere サーバ - SQL リファレンス』
 - 「DROP PROCEDURE 文」 『SQL Anywhere サーバ - SQL リファレンス』
 - 「DROP TABLE 文」 『SQL Anywhere サーバ - SQL リファレンス』
 - 「DROP TRIGGER 文」 『SQL Anywhere サーバ - SQL リファレンス』
 - 「DROP VIEW 文」 『SQL Anywhere サーバ - SQL リファレンス』
- **新しい INTO LOCAL TEMPORARY TABLE 句、SELECT 文** SELECT 文に新しく追加された INTO LOCAL TEMPORARY TABLE 句により、ローカル・テンポラリ・テーブルを作成し、SELECT 文の結果セットを設定できます。これまでは、テンポラリ・テーブル名が # から始まる場合に INTO 句を使用する方法しかありませんでした。「SELECT 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **新しい IF NOT EXISTS 句、CREATE TABLE 文** CREATE TABLE 文に新しく追加された IF NOT EXISTS 句により、永久テーブル、グローバル・テンポラリ・テーブル、ローカル・テンポラリ・テーブルが既存しない場合に作成できます。「CREATE TABLE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **テンポラリ・プロシージャやテンポラリ関数の作成時に所有者を指定** CREATE FUNCTION 文および CREATE PROCEDURE 文により、任意で所有者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。
 - 「CREATE FUNCTION 文 [Web サービス]」 『SQL Anywhere サーバ - SQL リファレンス』
 - 「CREATE PROCEDURE 文 [Web サービス]」 『SQL Anywhere サーバ - SQL リファレンス』

プログラマチック・インタフェース

次に、SQL Anywhere バージョン 11.0.1 でのプログラマチック・インタフェースの強化を示します。

- **新しい ASP.NET プロバイダ** 次の新しい ASP.NET プロバイダは、標準の ASP.NET プロバイダの機能を模倣しますが、データを SQL Server データベースではなく SQL Anywhere データベースに格納します。

- **メンバシップ** ログイン、ログアウト、ユーザおよびパスワードの管理を実現します。
- **ロール** ユーザのグループへの割り当て、および単純で簡単なパーミッション管理を実現します。
- **プロファイル** ユーザ変数を格納します。
- **Web パーツ・パーソナル化** Web パーツ・データの格納を管理し、ユーザによるビューのパーソナル化を実現します。
- **Web イベント** ヘルス・モニタリング機能と連携し、フラッシュされた Web イベント情報をロギング目的でデータベースに格納します。

「[SQL Anywhere ASP.NET プロバイダ](#)」 「[SQL Anywhere サーバ - プログラミング](#)」を参照してください。

- **新たに Ruby をサポート** SQL Anywhere は、Ruby オープン・ソース・プログラミング言語をサポートするようになりました。「[Ruby 用 SQL Anywhere](#)」 「[SQL Anywhere サーバ - プログラミング](#)」を参照してください。
- **OLE DB での CATALOGS と SCHEMATA の各ローセットのサポート** OLE DB の CATALOGS および SCHEMATA の各ローセットがサポートされるようになりました。SQL Anywhere はカタログの概念をサポートしていないため、SQL Anywhere OLE DB プロバイダは、CATALOGS の代わりに現在開始されているすべてのデータベースとそのロケーションを含む結果セットを返します。同様に SCHEMATA では、結果セット内でカタログとしてデータベース名が使用されます。

その他

次に、SQL Anywhere バージョン 11.0.1 でのその他の強化を示します。

- **単純で低コストな文の実行に関するパフォーマンスの向上** 実行時間が低コストの文のための実行プランを生成するのに要するコストは、文を実行するためのコストを上回る場合があります。次の強化により、SQL Anywhere で認識される、実行時間が低コストの単純な文のクラスが拡張され、このような文がオプティマイザをバイパスできるようになりました。

詳細については、「[クエリ処理のフェーズをスキップするための条件](#)」 「[SQL Anywhere サーバ - SQL の使用法](#)」を参照してください。

- **新しい FORCE NO OPTIMIZATION 句** 以前までは、コストベースの最適化が必要な文はすべてオプティマイザによって処理されていました。今回、FORCE NO OPTIMIZATION 句が追加されたことで、文がオプティマイザをバイパスするよう指定できるようになりました。文が複雑すぎて (データベース・オプションの設定またはスキーマやクエリの特徴などにより) このような処理が実行できない場合、文は失敗し、データベース・サーバはエラーを返します。

次の文は、新しい FORCE NO OPTIMIZATION 句をサポートしています。

- 「DELETE 文」 [『SQL Anywhere サーバ - SQL リファレンス』](#)
- 「INSERT 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』](#)
- 「UPDATE 文」 [『SQL Anywhere サーバ - SQL リファレンス』](#)

○ 新しい接続プロパティ

- QueryBypassedCosted
- QueryBypassedHeuristic
- QueryBypassedOptimized
- QueryOpened
- QueryDescribedBypass
- QueryDescribedOptimizer
- StatementDescribes
- StatementPostAnnotates
- StatementPostAnnotatesSimple
- StatementPostAnnotatesSkipped

○ 新しいデータベース・プロパティ

- Prepares
- QueryBypassedCosted
- QueryBypassedHeuristic
- QueryBypassedOptimized
- QueryOpened
- QueryDescribedBypass
- QueryDescribedOptimizer
- StatementDescribes
- StatementPostAnnotates
- StatementPostAnnotatesSimple
- StatementPostAnnotatesSkipped

- **新しい[最適化方法] フィールド** グラフィカルなプランの [オブティマイザ統計] セクションに、[最適化方法] フィールドが追加されました。このフィールドは、クエリ・オブティマイザによって選択された実行方式を返します。[「\[オブティマイザ統計\] フィールドの説明」](#) [『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・サーバがデフォルト・データベース・サーバにならないようにする** `-xd` サーバ・オプションにより、データベース・サーバがデフォルト TCP ポートで受信するのを防止し、データベース・サーバがデフォルト・データベース・サーバにならないようにします。[「-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。
- **並列アーカイブ・バックアップのサポート** SQL Anywhere データベース・サーバで、サーバ側のアーカイブ・バックアップに対する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、直列ではなく並列で読み書き操作を行うことで、パフォーマンスが向上します。

BACKUP DATABASE 文に、並列アーカイブ・バックアップをサポートするための新しい句が 2 つ追加されました。

- **WITH CHECKPOINT LOG [NO] COPY**
- **MAX WRITE { *n* | AUTO }**

バージョン 11.0.0 以前のデータベース・サーバでは、バージョン 11.0.1 のデータベース・サーバで生成されたアーカイブ・バックアップをリストアできません。バージョン 11.0.1 のデー

データベース・サーバでは、旧バージョンのデータベース・サーバで生成されたバックアップをリストアできます。

「[BACKUP 文](#)」 [『SQL Anywhere サーバ - SQL リファレンス』](#)を参照してください。

- **Mac OS X での Developer Edition と Evaluation Edition のデータベース・サーバの実行** Mac OS X で、Developer Edition と Evaluation Edition のデータベース・サーバを管理ツールから自動的に起動できるようになりました。

SQL Anywhere の動作の変更と廃止予定機能

次に、バージョン 11.0.1 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

- **全文検索** 次に、全文検索に加えられた動作の変更を示します。

- **演算子の優先度が適用される** 以前までは、クエリ文字列に演算子の優先度は適用されていませんでした。このリリースでは、次の演算子の優先度が適用されるようになりました。

- NEAR、FUZZY 演算子
- AND NOT 演算子
- AND 演算子
- OR 演算子

「CONTAINS 探索条件における演算子の優先度」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **NEAR 句の引数は単語またはプレフィクス単語にする** 近接検索を実行する場合、NEAR 句の引数は単語またはプレフィクス単語である必要があります。「CONTAINS 探索条件」『SQL Anywhere サーバ - SQL リファレンス』と「近接検索」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **ハイフンおよび AND NOT 句の使用** フレーズ内では、ハイフンは特殊文字ではなく単語区切りとして処理されます。フレーズの外にあるハイフンの処理は、ハイフンを囲む構文によって異なります。「ハイフン (-) を使用できる構文」『SQL Anywhere サーバ - SQL リファレンス』と「全文検索での AND NOT 演算子の使用」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **アスタリスクおよびプレフィクス検索の使用** プレフィクス検索を実行する場合、単語にアスタリスクを付加し、クエリ文字列の末尾とするか、またはそのすぐ後にスペースか許可された特殊文字を 1 つ挿入する必要があります。「アスタリスク (*) を使用できる構文」『SQL Anywhere サーバ - SQL リファレンス』と「プレフィクス検索」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **重複するテキスト・インデックスを作成するとエラーが返される** 重複するテキスト・インデックスを作成できなくなりました。次の設定が既存のテキスト・インデックスと同じ場合、そのテキスト・インデックスは重複するものと見なされます。
 - 参照されるベース・テーブル
 - インデックス対象カラム (順序は問わない)
 - 使用される設定オブジェクトの設定内容 (TERM BREAKER、MINIMUM TERM LENGTH、MAXIMUM TERM LENGTH、STOPLIST、照合情報)

SQL Anywhere 11.0.1 よりも前のバージョンで作成された重複するテキスト・インデックスは、データベース内に残すことができ、11.0.1 のデータベース・サーバで開始してもエ

ラーは発生しません。ただし、重複するテキスト・インデックスを含むデータベースをバージョン 11.0.1 以降で再ロードすると、エラーが返されます。

既存のデータベース内にある重複するテキスト・インデックスを識別するには、次のクエリを実行します。

```
SELECT LIST( i.index_name )
FROM SYS.SYSIDX i
      JOIN SYS.SYSTEXTIDX t ON i.object_id = t.index_id AND t.sequence = 1
      JOIN SYS.SYSTEXTCONFIG F ON F.object_id = t.text_config
      JOIN (
        SELECT table_id, index_id, LIST( column_id, ' ' ORDER BY column_id ) col_id
        FROM SYS.SYSIDXCOL
        GROUP BY table_id, index_id ) x
      ON x.table_id = i.table_id AND x.index_id = i.index_id
WHERE i.index_category=4
GROUP BY i.table_id, f.term_breaker, f.min_term_length, f.max_term_length,
       f.collation, ISNULL( f.char_stoplist, '-' ),
       ISNULL( f.nchar_stoplist, '-' ), x.col_id
HAVING count(*) > 1
```

このクエリは、ストップリストを表す文字列がまったく同じであるか、NO STOPLIST が指定されている場合のみ有効です。たとえば、ストップリスト 'a b c' とストップリスト 'a - b c' は、このクエリでは同じものとは見なされませんが、テキスト・インデックスを作成する際の重複のチェックでは同じものとは見なされます。

- **正規表現** SIMILAR TO、REGEXP の各探索条件、および REGEXP_SUBSTR 関数の動作に変更が加えられました。変更の主な目的は、SIMILAR TO の ANSI/SQL 標準との整合性を保ちながら、REGEXP と REGEXP_SUBSTR の動作を Perl と整合させることにあります。

- **データベースの照合および一致** 以前までは、REGEXP と REGEXP_SUBSTR は照合等価やソート順を使用して、パターン内のリテラル・クラスや文字クラスの範囲が文字列と一致するかを判断していました。このリリースでは、REGEXP と REGEXP_SUBSTR はコードポイント値のバイナリ比較を使用して、範囲の一致や評価を行っています。この変更は、Perl 5.0 と動作の整合性が保たれるようにするため行われています。

SIMILAR TO は依然としてデータベース照合を使用して一致や範囲の評価を行っています。「[LIKE 探索条件](#)、[REGEXP 探索条件](#)、[SIMILAR TO 探索条件](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。

- **データベースの大文字と小文字の区別および `[:upper:]` と `[:lower:]` の部分文字クラス** 大文字と小文字を区別しないデータベースでは、SIMILAR TO と REGEXP の `[:upper:]` と `[:lower:]` の部分文字クラスは区別されていませんでした。これは変更され、データベースが大文字と小文字を区別するかどうかにかかわらず、`[:upper:]` は大文字とだけ一致し、`[:lower:]` は小文字とだけ一致するようになりました。
- **メタ文字としての脱字記号 (^)、アンダースコア (_)、パーセント記号 (%) の処理** 次の表に、メタ文字としての各文字がどのように処理されるかを、以前の動作と新しい動作に分けて示します。

文字	以前の動作	新しい動作
<code>_</code> (アンダースコア)	SIMILAR TO、REGEXP、および REGEXP_SUBSTR では、アンダースコアはメタ文字として処理され、任意の 1 文字と一致していました。	SIMILAR TO では、アンダースコアはメタ文字として処理され、任意の 1 文字と一致します。 REGEXP と REGEXP_SUBSTR では、アンダースコアはメタ文字として処理されません。REGEXP と REGEXP_SUBSTR で任意の 1 文字と一致させるためには、代わりにピリオド (.) を使用します。
<code>%</code>	SIMILAR TO、REGEXP、および REGEXP_SUBSTR では、パーセント記号はメタ文字として処理され、任意の数の任意の文字と一致していました。	SIMILAR TO では、パーセント記号はメタ文字として処理され、任意の数の任意の文字と一致します。 REGEXP と REGEXP_SUBSTR では、パーセント記号はメタ文字として処理されません。REGEXP と REGEXP_SUBSTR で任意の数の任意の文字と一致させるためには、代わりにピリオドとアスタリスク (.*?) を使用します。

文字	以前の動作	新しい動作
^	SIMILAR TO、REGEXP、および REGEXP_SUBSTR では、文字クラス内の脱字記号は右にある文字の否定や減法として処理され、一致しないと解釈されていました。	SIMILAR TO では、脱字記号は右にある文字の否定や減法として処理されます。たとえば、SIMILAR TO [a-d^c] は a、b、d とは一致しますが、c とは一致しません。 REGEXP と REGEXP_SUBSTR では、脱字記号は文字クラス内で先頭に位置する場合のみメタ文字として処理され、文字クラスの否定として解釈されます。たとえば、REGEXP [^abc] は a、b、c 以外の任意の 1 文字と一致し、REGEXP [a-d^c] は a、b、c、d、および ^ と一致します。

- **Mac OS X で dbmodenv が不要になった** 以前のリリースでは、Mac OS X でグラフィカルな管理ツールを使用するためには、ユーザの `$HOME/.MacOSX/environment.plist` ファイルの `PATH` と `DYLD_LIBRARY_PATH` に、SQL Anywhere のバイナリとライブラリのロケーションを追加する必要がありました。そのために `dbmodenv` ツールを使用していました。SQL Anywhere は、`$HOME/.MacOSX/environment.plist` の設定に依存しなくなったため、`dbmodenv` を実行したり、SQL Anywhere をインストールした後にログアウトし、ログインし直す必要がなくなりました。
- **dbisqlc の OUTPUT 文から返される NULL 値のデフォルトが変更** 以前のリリースでは、`dbisqlc` の OUTPUT 文を使用すると、文は NULL 値として (NULL) 値を返していました。このリリースでは、NULL 値のデフォルトとして OUTPUT 文は空の文字列を返します。`output_nulls` オプションを設定することで、NULL 値のエクスポート方法を変更できます。[「output_nulls 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。
- **エンディアンのサポート** アップグレード後、ビッグ・エンディアン・コンピュータで作成された 11.0.1 よりも前のテキスト・インデックスは、トランケートおよびリフレッシュ (MANUAL REFRESH および AUTO REFRESH のテキスト・インデックスの場合)、または再作成 (IMMEDIATE REFRESH のインデックスの場合) が必要になります。
- **Mac OS X の管理ツール** Mac OS X の SQL Anywhere 管理ツールで、64 ビットの JDK 1.6 が使用されるようになりました。管理ツールは、Apple JDK 1.6 (Mac OS X 10.5.2 以降) でサポートされている、64 ビットのプロセッサを搭載した Intel Macintosh だけで動作します。Mac OS X 用の管理ツールを配備する場合、ネイティブ・ライブラリは `$SQLANY11/System/lib64` にあります。[「Linux、Solaris、Mac OS X における管理ツールの配備」](#) 『SQL Anywhere サーバ - プログラミング』を参照してください。

- **HTTP クライアントのチャンクされた転送コーディングの新しいデフォルト・サイズ** これまでは、HTTP クライアントから 2048 バイトを超えるデータが送信された場合、デフォルトで (またはユーザが `CREATE PROCEDURE ... SET 'HTTP(CH=auto)'` を指定した場合)、チャンクされた転送コーディングが試行されました。デフォルトのサイズが 2048 バイトから 8196 バイトに変更されました。また、チャンクされた転送コーディングを使用しないで要求を再発行する基準に、新しいステータス 411 **Length Required** が追加されました。「[CREATE PROCEDURE 文 \[Web サービス\]](#)」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

廃止予定機能とサポート終了機能

- **COMMENT ON EXTERNAL ENVIRONMENT OBJECT *object-name*** この構文は `COMMENT ON EXTERNAL OBJECT object-name` に変更されました。以前の構文は現在でも受け入れられますが、今後のリリースではサポートされない可能性があります。「[COMMENT 文](#)」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **ansi_substring オプションのサポート** ansi_substring オプションはバージョン 11.0.0 で非推奨になりましたが、バージョン 11.0.1 でまたサポートされるようになりました。「[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。

Mobile Link の新機能

次に、バージョン 11.0.1 で導入された Mobile Link の追加機能を示します。

- **mlsrv11 の -vi オプション** アップロードされた各ローのカラム値を表示します。「[-v オプション](#)」『[Mobile Link - サーバ管理](#)』を参照してください。
- **mlsrv11 の -vq オプション** ダウンロードされた各ローのカラム値を表示します。「[-v オプション](#)」『[Mobile Link - サーバ管理](#)』を参照してください。
- **mlsrv11 の -vm オプション** 同期が完了するたびに、各同期と各同期フェーズの継続時間をログに出力します。「[-v オプション](#)」『[Mobile Link - サーバ管理](#)』を参照してください。
- **mlsrv11 の -ppv オプション** 指定された期間に従って、Mobile Link が新しい定期モニタリングの値を出力するようにします。「[-ppv オプション](#)」『[Mobile Link - サーバ管理](#)』を参照してください。
- **ml_ignore プレフィクス** Mobile Link サーバでは、--{ml_ignore} で始まる SQL スクリプトが、意図的に無視するスクリプトとして認識されます。「[スクリプトの無視](#)」『[Mobile Link - サーバ管理](#)』を参照してください。
- **dblsn の -sv オプション** データベースを認証するために Mobile Link Listener が使用するスクリプト・バージョンを指定します。「[-sv オプション](#)」『[Mobile Link - サーバ起動同期](#)』を参照してください。
- **Oracle VArray のサポート** iAnywhere Solutions 11 - Oracle ODBC ドライバで、ストアード・プロシージャでの Oracle VArray の使用がサポートされるようになりました。「[Oracle varray の使用](#)」『[Mobile Link - サーバ管理](#)』を参照してください。
- **ライトウェイト・ポーリングのリスナ・キーワード変数** ライトウェイト・ポーリングをサポートする、poll_connect、poll_notifier、poll_key、poll_every の各リスナ・キーワードが追加されました。
- **ライトウェイト・ポーリングのアクション変数** ライトウェイト・ポーリングをサポートする、\$poll_connect、\$poll_notifier、\$poll_key、\$poll_every の各アクション変数が追加されました。
- **Common Access Card を使用したクライアント認証** Mobile Link クライアントで、Common Access Card (CAC) のクライアント ID を使用した認証がサポートされるようになりました。「[identity_name](#)」『[Mobile Link - クライアント管理](#)』を参照してください。

別途ライセンスが必要な必須コンポーネント

この機能は CAC 認証アドオンの一部であり、別途ライセンスが必要です。「[別途ライセンスが必要なコンポーネント](#)」『[SQL Anywhere 11 - 紹介](#)』を参照してください。

- **Microsoft SQL Server 2008 のサポート** Mobile Link 同期サーバで、Microsoft SQL Server 2008 で実行されている統合データベースがサポートされるようになりました。Microsoft の新しい DATE、TIME、DATETIME2 の各データ型のマッピングについては、「[Microsoft SQL Server データのマッピング](#)」『[Mobile Link - サーバ管理](#)』を参照してください。

- **.NET DownloadTableData インタフェースの新しいメソッド** `getLastDownloadTime` メソッドによってテーブルの最終ダウンロード時刻が返されます。「[GetLastDownloadTime メソッド](#)」『[Mobile Link - サーバ管理](#)』を参照してください。
- **対象 Mobile Link ユーザとリモート ID に対する ログの冗長性** 対象とする Mobile Link ユーザまたはリモート ID に異なるログの冗長性を設定できるようになりました。「[対象 Mobile Link ユーザとリモート ID に対する ログの冗長性](#)」『[Mobile Link - サーバ管理](#)』を参照してください。

Mobile Link の動作の変更と廃止予定機能

次に、バージョン 11.0.1 で導入された Mobile Link の変更点を示します。

- **mlsrv11 の -sm オプション** mlsrv11 の -sm オプションが改善され、非永続的な HTTP/HTTPS に使用したときに -nc オプションに似た機能を提供するようになりました。「[-sm オプション](#)」『[Mobile Link - サーバ管理](#)』と「[-nc オプション](#)」『[Mobile Link - サーバ管理](#)』を参照してください。
- **Microsoft SQL Server データ型** SQL Server 2005 で、TEXT、NTEXT、IMAGE の各データ型が推奨されなくなりました。代わりに VARCHAR(max)、NVARCHAR(max)、VARBINARY(max) を使用してください。「[Microsoft SQL Server データのマッピング](#)」『[Mobile Link - サーバ管理](#)』を参照してください。

QAnywhere の新機能

次に、バージョン 11.0.1 で導入された QAnywhere の追加機能を示します。

- **QAnywhere スタンドアロン・クライアント** QAnywhere スタンドアロン・クライアントは、QAnywhere Agent を実行したり、データベースを管理したりせずに、メッセージング・システムを設定できる小型のクライアントです。[「QAnywhere スタンドアロン・クライアントの概要」](#) [『QAnywhere』](#) を参照してください。

Ultra Light の新機能

次に、バージョン 11.0.1 で導入された Ultra Light の追加機能を示します。

- **ミラー・ファイル** Ultra Light には、潜在的に信頼性が低いストレージ・システムのフォールト・トレランスを改善する基本的なデータベース・ファイル・ミラーリング機能があります。ミラーリングには、ミラー・ファイルが使用されます。データベースへのすべての書き込みは、メイン・データベース・ファイルに対して発行されると同時にミラー・ファイルにも発行されます。「[Ultra Light MIRROR_FILE 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。
- **ml_remote_id の設定解除** SET OPTION を使用して **ml_remote_id** の設定を解除できるようになりました。「[Ultra Light SET OPTION 文](#)」『[Ultra Light データベース管理とリファレンス](#)』と「[ML_GET_SERVER_NOTIFICATION \[システム\]](#)」『[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。
- **ML_GET_SERVER_NOTIFICATION** この関数を使用すると、Ultra Light ユーザはライトウェイト・ポーリングを使用して、サーバ起動同期要求について Mobile Link サーバ上の Notifier に問い合わせできます。「[ML_GET_SERVER_NOTIFICATION \[システム\]](#)」『[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。
- **SYNC_PROFILE_OPTION_VALUE 関数 [システム]** この関数は、同期プロファイル内の指定されたオプションの値を返します。「[SYNC_PROFILE_OPTION_VALUE 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。
- **Ultra Light .NET API の StreamErrorParameters プロパティ** StreamErrorParameters が SyncResult クラスに追加されました。このメンバには、StreamErrorCode でレポートされるストリーム・エラー・コードのエラー・パラメータをカンマで区切ったリストが含まれます。「[StreamErrorParameters 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 M-Business API の getStreamErrorParameters メソッド** このメソッドは、同期ストリーム処理によってレポートされるエラー・パラメータをカンマで区切ったリストを返します。「[getStreamErrorParameters 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。

Ultra Light の動作の変更と廃止予定機能

次に、バージョン 11.0.1 で導入された Ultra Light の変更点を示します。

● **Palm のサスペンド機能** 次の項を参照してください。

- 「Ultra Light Palm アプリケーションのステータスの管理 (旧式)」 『Ultra Light - C/C++ プログラミング』
- 「Ultra Light Palm アプリケーションの状態のリストア (旧式)」 『Ultra Light - C/C++ プログラミング』
- 「GetSuspend 関数 (旧式)」 『Ultra Light - C/C++ プログラミング』
- 「SetSuspend 関数 (旧式)」 『Ultra Light - C/C++ プログラミング』
- 「GetSuspend 関数 (旧式)」 『Ultra Light - C/C++ プログラミング』
- 「SetSuspend 関数 (旧式)」 『Ultra Light - C/C++ プログラミング』
- 「SQLSetSuspend 関数 (旧式)」 『Ultra Light - C/C++ プログラミング』

Ultra Light J の新機能

次に、バージョン 11.0.1 で導入された Ultra Light J の追加機能を示します。

- **追加 SQL サポート** Ultra Light J には、次の追加 SQL サポートがあります。

- ALTER TABLE
- CREATE TABLE
- CREATE INDEX
- DROP INDEX
- DROP TABLE
- TRUNCATE TABLE

また、次の制約がなくなりました。

- HAVING がサポートされます。
- 集合関数内の DISTINCT がサポートされます。
- CURRENT TIME、CURRENT TIMESTAMP、CURRENT DATE がサポートされます。

「サポートされている SQL 文」『Ultra Light J』を参照してください。

- **DatabaseInfo の新しいメソッド** `getPageReads()` と `getPageWrites` の 2 つの新しいメソッドが `DatabaseInfo` インタフェースに追加されました。これらのメソッドは、`DatabaseInfo` オブジェクトの作成時点のページ読み込みとページ書き込みの数を返します。「[getPageReads メソッド](#)」『Ultra Light J』と「[getPageWrites メソッド](#)」『Ultra Light J』を参照してください。
- **Ultra Light J データベース転送ユーティリティの更新** Ultra Light J データベース転送ユーティリティで、クライアントからのデータベースの転送に加えて、データベースの削除、データベース情報の表示、データベース転送ログの表示または電子メール送信が可能になりました。「[Ultra Light J データベース転送ユーティリティ \(ULjDbT\)](#)」『Ultra Light J』を参照してください。

Ultra Light J の動作の変更と廃止予定機能

次に、バージョン 11.0.1 で導入された Ultra Light J の変更点を示します。

- **@@identity グローバル変数** @@identity グローバル変数は Ultra Light J でサポートされていません。

管理ツールの新機能

次に、バージョン 11.0.1 で導入された Sybase Central と Interactive SQL の追加機能を示します。

Sybase Central の新機能

次に、バージョン 11.0.1 で導入された Sybase Central のプラグインの追加機能を示します。

すべてのプラグイン

- **サービス作成ウィザードの強化** サービス作成ウィザードを使用して、Mobile Link リレー・サーバ(rshost)、Mobile Link リレー・サーバ Outbound Enabler (RSOE)、SQL Anywhere ポリユー・ム・シャドウ・コピー・サービス (dbvss11)、Mobile Link Listener ユーティリティ (dblsn)、Broadcast Repeater ユーティリティ (dbns11) のサービスを作成できるようになりました。[「Windows サービスとして実行できるプログラム」](#) [『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

- **Sybase Central 内でイベントのステータスを表示** Sybase Central で、イベントの現在の実行ステータスを確認できるようになりました。[イベント] フォルダの新しい **[実行中]** カラムに、イベントの現在の実行ステータスが表示されます。左ウィンドウ枠でフォルダを選択すると、フォルダの内容が更新されます。また、[イベントのプロパティ] ウィンドウに **[実行中]** プロパティが表示されます。

この値は、イベントが実行中の場合は **[はい]**、イベントが実行中ではない場合は **[いいえ]**、イベントが SQL Anywhere 9.0.2 以前のデータベースにある場合は **[不明]** になります。
- **Sybase Central 内でのメンテナンス・プランのステータスの表示** Sybase Central で、メンテナンス・プランの現在の実行ステータスを確認できるようになりました。[メンテナンス・プラン] フォルダの新しい **[実行中]** カラムに、プランの現在の実行ステータスが表示されます。左ウィンドウ枠でフォルダを選択すると、フォルダの内容が更新されます。

この値は、メンテナンス・プランが実行中の場合は **[はい]**、プランが実行中ではない場合は **[いいえ]**、プランが SQL Anywhere 9.0.2 以前のデータベースにある場合は **[不明]** になります。
- **メンテナンス・プランの強化** メンテナンス・プラン作成ウィザードで、メンテナンス・プラン・レポートを電子メールで送信する設定を行う際に、テスト電子メールを送信できるようになりました。[「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。
- **プロシージャ作成ウィザードとファンクション作成ウィザードの強化** ファンクション作成ウィザードとプロシージャ作成ウィザードで、プロシージャまたは関数を記述する SQL ダイアレクトまたは言語を、Watcom-SQL、Transact-SQL、外部 C/C++、または外部環境から選択できるようになりました。外部環境を選択した場合、C_ESQL32、C_ESQL64、C_ODBC32、C_ODBC64、CLR、JAVA、PERL、PHP のいずれかの言語を選択します。これまでは、Watcom-SQL または Transact-SQL だけを選択できました。C/C++ または Java を選択すると、EXTERNAL NAME 句を含む関数またはプロシージャのコード・スケルトンが生成されます。[「CREATE FUNCTION 文 \[外部プロシージャ\]」](#) [『SQL Anywhere サーバ - SQL リファレンス』](#)

と「[CREATE PROCEDURE 文 \[外部プロシージャ\]](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。

Mobile Link プラグイン

- **Oracle データベース用の Mobile Link モデルの強化** 同期モデル作成ウィザードを使用して Oracle 統合データベースを使用する同期モデルを作成するときに、スキーマ全体をロードするか、同期に必要なテーブルの所有者のサブセットを選択するかを選べるようになりました。所有者のサブセットを選択すると、スキーマのロード時間を短縮できます。

管理ツールの動作の変更と廃止予定機能

次に、バージョン 11.0.1 で導入された Sybase Central と Interactive SQL の変更点を示します。

Sybase Central の動作の変更と廃止予定機能

次に、バージョン 11.0.1 で導入された Sybase Central の変更点を示します。

SQL Anywhere プラグイン

- **読み込み専用データベースの強化** 読み込み専用データベースに接続するときに警告が表示されるようになりました。この警告が表示されないようにするには、[ツール] - [SQL Anywhere 11] - [ユーザ設定] を選択して設定します。

Interactive SQL の動作の変更と廃止予定機能

次に、バージョン 11.0.1 で導入された Interactive SQL の変更点を示します。

- **データベース・ロックの自動解放の設定** 結果セットが表示されるときに作成されるデータベース・スキーマ・ロックを解放するように Interactive SQL を設定できるようになりました。そのように設定するには、Interactive SQL で [ツール] - [オプション] - [SQL Anywhere] を選択し、[データベース・ロックの自動解放] を選択します。

このオプションを選択すると、結果セットを返す文を実行した後、データベースにコミットされていない変更が接続にあるかどうか Interactive SQL によって確認されます。コミットされていない変更がなかった場合は Interactive SQL によってスキーマ・ロックが解放されます。それ以外の場合、スキーマ・ロックは解放されません。つまり、データベースにコミットされていない変更がある場合、スキーマ・ロックは解放されません。

製品全体の新機能

次に、バージョン 11.0.1 で導入された製品全体の追加機能を示します。

● 新しいオンライン・マニュアル・フォーラム、DocCommentXchange

(DCX) DocCommentXchange と呼ばれる新しいオンライン・フォーラムが作成されました。DocCommentXchange は、SQL Anywhere マニュアルにアクセスし、マニュアルについて議論するためのコミュニティです。

DocCommentXchange は次の目的に使用できます (現在のところ、日本語はサポートされていません)。

- マニュアルを表示する
- マニュアルの項目について明確化するために、ユーザによって追加された内容を確認する
- すべてのユーザのために、今後のリリースでマニュアルを改善するための提案や修正を行う

<http://dcx.sybase.com> を参照してください。

● SQL Anywhere モニタ SQL Anywhere モニタは、SQL Anywhere データベースや Mobile Link サーバの正常性や可用性に関する情報を示す、ブラウザベースの管理ツールです。

Mobile Link では、この機能は既存の Mobile Link モニタの機能に代わるものではなく、また Mobile Link モニタの機能と重複しません。

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

マニュアルの強化

- **SQL Anywhere のバックアップとリカバリのマニュアルの強化** SQL Anywhere に付属のバックアップ・ツールおよびリカバリ・ツールを使用するためのマニュアルが、今回のリリースに合わせて書き直されました。「[バックアップとデータ・リカバリ](#)」『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。
- **全文検索のマニュアルの強化** 全文検索機能のマニュアルの構成が変更され、例やチュートリアルが追加されました。「[全文検索](#)」『[SQL Anywhere サーバ - SQL の使用法](#)』を参照してください。
- **Mobile Link サーバ起動同期のマニュアル** Mobile Link サーバ起動同期のマニュアルが、バージョン 11.0.0 から改善されました。これには、目次の構成の改善、サーバ起動同期の説明の詳細化、リファレンス情報のフォーマットの改善などが含まれます。[Mobile Link - サーバ起動同期](#)を参照してください。
- **SQL Remote のマニュアル** SQL Remote のマニュアルは、構成の変更と改訂が行われ、使いやすくなりました。[SQL Remote](#) を参照してください。

バージョン 11.0.0 の新機能

目次

SQL Anywhere	29
Mobile Link	72
QAnywhere	78
SQL Remote	80
Ultra Light	81
Sybase Central と Interactive SQL	88
マニュアルの強化	98
製品全体の機能	99

SQL Anywhere のバージョン 8 より前における新機能や動作の変更については、http://www.iAnywhere.com/developer/product_manuals/sqlanywhere/1001/en/html/dbwnen10/dbwnen10.html を参照してください。

SQL Anywhere

- 「SQL Anywhere の新機能」 29 ページ
- 「SQL Anywhere の動作の変更」 58 ページ
- 「SQL Anywhere の廃止予定機能とサポート終了機能」 68 ページ

Mobile Link

- 「Mobile Link の新機能」 72 ページ
- 「Mobile Link の動作の変更と廃止予定機能」 76 ページ

QAnywhere

- 「QAnywhere の新機能」 78 ページ
- 「QAnywhere の動作の変更と廃止予定機能」 79 ページ

SQL Remote

- 「SQL Remote の新機能」 80 ページ
- 「SQL Remote の動作の変更と廃止予定機能」 80 ページ

Ultra Light

- 「Ultra Light の新機能」 81 ページ
- 「Ultra Light の動作の変更と廃止予定機能」 86 ページ

Sybase Central と Interactive SQL

- 「Sybase Central と Interactive SQL の新機能」 88 ページ
- 「Sybase Central と Interactive SQL の動作の変更と廃止予定機能」 91 ページ

マニュアルの強化

- 「マニュアルの強化」 98 ページ

廃止予定機能は変更される可能性があります

廃止予定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。

製品全体の機能

- 「製品全体の新機能」 99 ページ
- 「製品全体の動作の変更」 99 ページ

SQL Anywhere

次の項では、SQL Anywhere バージョン 11.0.0 の新機能、動作の変更、廃止予定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。

SQL Anywhere の新機能

次に、バージョン 11.0.0 で導入された SQL Anywhere データベースとデータベース・サーバの追加機能を示します。

主な機能

次に、SQL Anywhere バージョン 11.0.0 の主な機能を示します。

- **テーブルのマージのサポート** SQL Anywhere では、テーブル、ビュー、システム・プロシージャの結果をテーブルまたはビューにマージできるようになりました。[「MERGE 文」](#) [『SQL Anywhere サーバ - SQL リファレンス』](#) を参照してください。
- **ログイン・ポリシーのサポート** SQL Anywhere では、ログイン・ポリシーがサポートされるようになりました。ログイン・ポリシーは、ユーザに対してデータベース接続が確立される時に適用するルールを定義する一連のオプションです。SQL Anywhere には、すべてのログイン・ポリシーのデフォルト値を含むルート・ポリシーがあります。ルート・ログイン・ポリシーに優先する設定を含む個別のログイン・ポリシーを作成できます。ログイン・ポリシーは各ユーザに個別に割り当てます。各ユーザが所属するグループからログイン・ポリシーを継承することはありません。特定のユーザへのポリシーの割り当ては、必要に応じて変更できます。[「ログイン・ポリシーの管理の概要」](#) [『SQL Anywhere サーバ - データベース管理』](#) を参照してください。

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。[「SQL Anywhere のアップグレード」 392 ページ](#)を参照してください。

- **全文検索のサポート** SQL Anywhere で全文検索がサポートされるようになりました。全文検索を行うと、データベース内で特定の単語が出現するすべての箇所を簡単に検索できます。全文検索は、単語単位であるという点、またテーブルのローをスキャンするのではなくテキスト・インデックスを使用する点で、LIKE、REGEXP、SIMILAR TO などの述部を使用した検索とは異なります。[「全文検索」](#) [『SQL Anywhere サーバ - SQL の使用法』](#) を参照してください。

全文検索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。[「SQL Anywhere のアップグレード」 392 ページ](#)を参照してください。

- **正規表現のサポート** SQL Anywhere では、REGEXP と SIMILAR TO の 2 つの探索条件を使用した正規表現がサポートされるようになりました。[「REGEXP 探索条件」](#) [『SQL](#)

[Anywhere サーバ - SQL リファレンス](#)』と「SIMILAR TO 探索条件」 [『SQL Anywhere サーバ - SQL リファレンス』](#)を参照してください。

「正規表現の概要」 [『SQL Anywhere サーバ - SQL リファレンス』](#)と「LIKE 探索条件、REGEXP 探索条件、SIMILAR TO 探索条件」 [『SQL Anywhere サーバ - SQL リファレンス』](#)も参照してください。

- **データベース・オプション設定のトランザクション・ログへの記録** LOAD 操作時に有効なデータベース・オプション設定がトランザクション・ログに記録されるようになりました。このため、最初の LOAD 操作と、リカバリ時にトランザクション・ログを適用して実行される最後の LOAD 操作で、date_order や nearest_century などのオプションが違って、データの矛盾が生じません。
- **オプティマイザでのインデックス使用の強化** SQL Anywhere のインデックス機能がいくつかの点で強化されました。これらの新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。「[SQL Anywhere のアップグレード](#)」 392 ページを参照してください。
 - **複合インデックス・スキャンのサポート** オプティマイザが強化され、複数のインデックス (最大 4 つまで) を考慮して、ベース・テーブルの複数の述部に基いてテーブルからデータが取り出されるようになりました。これまでは、クエリのインデックス・ヒントとしてインデックスを 1 つだけ指定できました。SELECT 文の WITH 句の新しいインデックス・ヒントを使用すると、複合インデックス・スキャンを使用可能なことを指定できます。「FROM 句」 [『SQL Anywhere サーバ - SQL リファレンス』](#)を参照してください。

複合インデックス・スキャンという新しいテーブル・アクセス・アルゴリズムが追加されました。「[MultipleIndexScan 方式 \(MultIdx\)](#)」 [『SQL Anywhere サーバ - SQL の使用法』](#)を参照してください。
 - **インデックス専用取得のサポート** オプティマイザが強化され、インデックス専用取得がサポートされるようになりました。インデックス専用取得では、テーブル内の対応するローにアクセスする必要がなく、インデックスのデータを使用してクエリが処理されます。オプティマイザは、可能な場合は常にインデックス専用取得を実行します。INDEX ONLY {ON|OFF} ヒントを使用して、インデックス専用取得を実行するかどうかを制御できます。「[インデックスを使用して述部を満たすことができる](#)」 [『SQL Anywhere サーバ - SQL の使用法』](#)と「FROM 句」 [『SQL Anywhere サーバ - SQL リファレンス』](#)を参照してください。
- **データのロードとアンロードの強化** データのロードとアンロードが次のように強化されます。
 - **クライアント・コンピュータにあるファイルからのデータのロードとファイルへのデータのアンロード** データベース・サーバでは、これまでも SQL 文や関数を使用して、データベース・サーバのコンピュータにあるデータの読み込みと書き込みがサポートされていました。今回実装された新しい機能では、この機能をクライアント・コンピュータにあるファイルにも適用できるようになったので、クライアントのファイルをデータベース・サーバのコンピュータにコピーする必要がなくなりました。このデータの転送は効率的に行われる一方、クライアント・コンピュータ上のデータに対するセキュリティとアクセス制御が実現します。

クライアント・コンピュータにあるファイルの実際の読み込みはクライアント・ライブラリによって透過的に行われます。したがって、既存のクライアント・アプリケーションで新しい SQL 言語のサポートを使用することで、この新機能をすぐに利用できます。

この新機能を利用するには、クライアントとデータベース・サーバの両方が SQL Anywhere バージョン 11.0.0 であり、またクライアントで Command Sequence 通信プロトコル (CmdSeq) を使用する必要があります。

「クライアント・コンピュータ上のデータへのアクセス」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **変数へのデータのアンロード** UNLOAD 文が強化されて INTO VARIABLE 句が追加され、変数にデータをアンロードできるようになりました。「UNLOAD 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **別のテーブルのカラムからのデータのロード** LOAD TABLE 文が強化されて USING COLUMN 句が追加され、別のテーブルのカラムからデータをロードできるようになりました。「LOAD TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』と「LOAD TABLE 文を使用したデータのインポート」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

この新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。「SQL Anywhere のアップグレード」 392 ページを参照してください。

- **値 (BLOB) からのデータのロード** LOAD TABLE 文が強化されて USING VALUE 句が追加され、関数やシステム・プロシージャの結果などの値式からデータをロードできるようになりました。「LOAD TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』と「LOAD TABLE 文を使用したデータのインポート」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

この新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。「SQL Anywhere のアップグレード」 392 ページを参照してください。

- **LOAD TABLE 文のリカバリとミラーリングの強化** これまでは、ミラーリングされたデータベース構成で LOAD TABLE 文を使用してファイルからデータをロードできませんでした。これは、LOAD TABLE 文だけがトランザクション・ログに記録され、ロード対象のデータは記録されなかったからです。また、データベースをリカバリするときに、LOAD TABLE 文を使用してロードされたデータは、リカバリ時に元のロード・ファイルがないかぎり回復不能でした。

LOAD TABLE 文が強化され、WITH CONTENT LOGGING、WITH ROW LOGGING、WITH FILE NAME LOGGING の 3 つの新しいログ・オプション句が追加されました。これらの句を使用すると、ロードするデータをトランザクション・ログに記録するかどうかを制御できます。データベースのミラーリング・システムでは、このデータを使用してミラー・データベースをロードできます。また、リカバリ時にロード・ファイルがなくてもリカバリが可能になります。「LOAD TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

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。「SQL Anywhere のアップグレード」 392 ページを参照してください。

- **マテリアライズド・ビューの強化** マテリアライズド・ビューのサポートが次のように強化されました。
- **即時マテリアライズド・ビューのサポート** 基本となるテーブル内のデータの変更がマテリアライズド・ビュー内のデータに影響する場合に、マテリアライズド・ビューをすぐにリフレッシュするように設定できます。このようにすぐにリフレッシュするビューを「即

時ビュー」と呼び、すぐにリフレッシュされないビューは今後「手動ビュー」と呼びます。このリリースの前に作成されたマテリアライズド・ビューは手動ビューと見なされます。また、新規に作成するマテリアライズド・ビューはデフォルトで手動ビューになります。

手動ビューと即時ビューの詳細については、「[マテリアライズド・ビューの操作](#)」
『[SQL Anywhere サーバ - SQL の使用法](#)』と「[手動マテリアライズド・ビューと即時マテリアライズド・ビュー](#)」
『[SQL Anywhere サーバ - SQL の使用法](#)』を参照してください。

この機能をサポートするシステム・プロシージャを使用するには、データベースをアップグレードする必要があります。「[SQL Anywhere のアップグレード](#)」 392 ページを参照してください。

- **複数のマテリアライズド・ビューの同時リフレッシュ** これまでは、マテリアライズド・ビューは一度に 1 つずつリフレッシュする必要がありました。各リフレッシュ操作の間で基本となるデータが変更された場合、マテリアライズド・ビューの間で矛盾が生じる可能性があります。このリリースでは、同じデータを使用するマテリアライズド・ビューをリフレッシュするときに、REFRESH MATERIALIZED VIEW 文にマテリアライズド・ビューのリストを指定できるようになりました。「[REFRESH MATERIALIZED VIEW 文](#)」
『[SQL Anywhere サーバ - SQL リファレンス](#)』と「[手動ビューのリフレッシュ](#)」
『[SQL Anywhere サーバ - SQL の使用法](#)』を参照してください。
- **REFRESH MATERIALIZED VIEW 文の新しい WITH SHARE MODE 句** REFRESH MATERIALIZED VIEW 文に、新しい句 WITH SHARE MODE が追加されました。このモードでは、リフレッシュ操作中に、基本となるテーブルを他のトランザクションで読み込むことができます。この句を指定すると、基本となるすべてのテーブルの共有テーブル・ロックが取得されてから、リフレッシュ操作が実行されます。マテリアライズド・ビューが IMMEDIATE REFRESH と定義されているか、データベースのスナップショット・アイソレーションが有効になっていないかぎり、デフォルトのモードは WITH SHARE MODE になります。リフレッシュのデフォルト動作の詳細については、「[REFRESH MATERIALIZED VIEW 文](#)」
『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。
- **ファイルまたは BLOB 文字列の内容の問い合わせのサポート** FROM 句の新しい OPENSTRING 句を使用して、ファイルや BLOB 文字列のデータを問い合わせることができるようになりました。OPENSTRING 句では、問い合わせるオブジェクト、またデータのスキーマやその他の解析情報を指定できます。「[FROM 句](#)」
『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。

OPENSTRING 操作が実行されると、新しいプラン項目 OpenString が実行プランに表示されません。「[OpenString アルゴリズム \(OpenString\)](#)」
『[SQL Anywhere サーバ - SQL の使用法](#)』を参照してください。
- **圧縮インデックスのサポートの向上** 圧縮インデックスのサポートが向上された結果、データベースをアンロードしてから再ロードして再構築すると、再構築されたデータベースが元のデータベースよりも小さくなる場合があります。このデータベースのサイズの縮小は、問題や、データが失われたことを示すものではありません。
- **ミラー・サーバで実行されているデータベースへの読み込み専用アクセス** データベース・ミラーリングを使用している場合に、ミラー・サーバで実行されているデータベースに接続できるようになりました。したがって、プライマリ・サーバの可用性を維持しながら、リソー

スを大量に使用する可能性があるレポート操作の負荷をミラー・サーバに割り当てることができます。ミラー・データベースに接続するには、読み込み専用のミラー・データベースへのアクセスに使用できる `-sm` サーバ・オプションでデータベース・サーバ名を指定します。「[-sm 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「[ミラー・サーバで実行されているデータベースへの読み込み専用アクセスの設定](#)」『[SQL Anywhere サーバ-データベース管理](#)』を参照してください。

データベース接続

次に、SQL Anywhere バージョン 11.0.0 でのデータベース接続の強化を示します。

- **AppInfo 接続パラメータの強化** AppInfo 接続パラメータで OSUSER キーがサポートされるようになりました。このキーは、クライアント・プロセスに関連付けられたオペレーティング・システム・ユーザ名を返します。Linux と Solaris で EXE キーがサポートされるようになりました。「[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。
- **Elevate 接続パラメータ** Elevate 接続パラメータは、Windows Vista で自動的に起動した SQL Anywhere データベース・サーバを昇格します。「[Elevate 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。
- **NewPassword 接続パラメータ [NEWPWD]** NewPassword 接続パラメータを使用すると、DBA に依頼しなくてもユーザが各自でパスワードを変更できます。パスワードの有効期限が切れていてもかまいません。「[NewPassword 接続パラメータ \[NEWPWD\]](#)」『[SQL Anywhere サーバ-データベース管理](#)』を参照してください。
- **プリフェッチの強化** PrefetchBuffer (PBUF) 接続パラメータのデフォルト値が変更されました。デフォルト値は、Windows Mobile では 64 KB、その他のプラットフォームでは 512 KB になりました。この接続パラメータには、64 KB ～ 8 MB の値を指定できます。「[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。

これまでのバージョンでは、プリフェッチされる最大ロー数は、プリフェッチできる最大データ量に基づいていました。このリリースでは、プリフェッチされる最大ロー数には、プリフェッチされる実際のデータ量に加えて、PrefetchBuffer 接続パラメータで指定されたデータ制限が考慮されるようになりました。その結果、カラム内のデータ量が、ホスト変数長と記述長の両方よりも大幅に少ない場合に、パフォーマンスが大きく向上する可能性があります。

また、パフォーマンスが向上する可能性が高い場合に、プリフェッチするロー数が動的に増えます。「[ローのプリフェッチ](#)」『[SQL Anywhere サーバ-プログラミング](#)』を参照してください。

バックアップとリカバリ

次に、SQL Anywhere バージョン 11.0.0 でのバックアップとリカバリの強化を示します。

- **Microsoft ボリューム・シャドウ・コピー・サービス (VSS) のサポート** SQL Anywhere は、Microsoft ボリューム・シャドウ・コピー・サービス (VSS) との互換性があります。VSS を使用するには、既存のデータベースをすべて再構築する必要があります。「[SQL Anywhere ボ](#)

リウム・シャドウ・コピー・サービス (VSS) 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。

セキュリティ

次に、SQL Anywhere バージョン 11.0.0 でのセキュリティの強化を示します。

- **テーブルの暗号化が有効になっているときの ISYSUSER と ISYSEXTERNLOGIN の各システム・テーブルの暗号化** これまでは、データベースを暗号化するときや、テーブルの暗号化を有効にしてデータベースを作成するときに、ISYSCOLSTAT システム・テーブルが自動的に暗号化されました。このリリースでは、さらにセキュリティを強化するため、ISYSUSER と ISYSEXTERNLOGIN の各システム・テーブルも暗号化されるようになりました。

- **監査の強化** このリリースでは、Sybase Central を使用して監査を制御できるようになりました。DBA 権限を持つユーザは、[データベースのプロパティ] ウィンドウから、監査を有効または無効にしたり、監査する情報を指定したりできます。監査情報は、Sybase Central の右ウィンドウ枠の [監査] タブで確認できます。「監査の制御」『SQL Anywhere サーバ - データベース管理』と「監査情報の取り出し」『SQL Anywhere サーバ - データベース管理』を参照してください。

監査が有効になっているときは、接続失敗のエラーが、失敗の理由とともにログに記録されます。

- **256 ビットの AES 暗号化のサポート** SQL Anywhere で、データベース、テーブル、ファイル、データに対して 256 ビットの AES 暗号化がサポートされるようになりました。この強化は、次に示す複数の領域に影響します。

- **データベースとテーブルの暗号化** CREATE DATABASE 文の ENCRYPTION 句に、AES256 と AES256_FIPS を指定できるようになりました。「CREATE DATABASE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

初期化ユーティリティ (dbinit) とアンロード・ユーティリティ (dbunload) の -ea オプションに、AES256 と AES256_FIPS を指定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「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。

- **FIPS 認定アルゴリズム** FIPS 対応のプラットフォームで、256 ビットの FIPS 認定 AES アルゴリズムを使用できるようになりました。「-fips 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。

- **データの暗号化と復号化** ENCRYPT 関数と DECRYPT 関数を使用してデータを暗号化するときに、AES256 と AES256_FIPS を指定できるようになりました。「ENCRYPT 関数 [文字列]」『SQL Anywhere サーバ - SQL リファレンス』と「DECRYPT 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、トランザクション・ログ、DB 領域の暗号化されたコピーの作成** CREATE ENCRYPTED FILE 文を使用して、暗号化されているか、暗号化されていないデータベース、トランザクション・ログ、または DB 領域の暗号化されたコピーを作成するとき、256 ビットの AES アルゴリズム (AES256 または AES256_FIPS) を指定できるようになりました。

た。「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。

- **DBTools での 256 ビットの AES 暗号化のサポート** a_create_db と an_unload_db の各構造体が拡張され、encryption_algorithm メンバの値として AES256 と AES256_FIPS がサポートされるようになりました。「a_create_db 構造体」『SQL Anywhere サーバ - プログラミング』と「an_unload_db 構造体」『SQL Anywhere サーバ - プログラミング』を参照してください。

参照

- 「データベースの暗号化と復号化」『SQL Anywhere サーバ - データベース管理』
- 「データベース・プロパティ」『SQL Anywhere サーバ - データベース管理』
- **jConnect と Open Client のパスワード暗号化のサポート** jConnect 接続と Open Client 接続でパスワードの暗号化がサポートされるようになりました。次の項を参照してください。
 - 「jConnect JDBC ドライバの使用」『SQL Anywhere サーバ - プログラミング』
 - 「JDBC クライアントの配備」『SQL Anywhere サーバ - プログラミング』
 - 「SQL Anywhere における Open Client の既知の制限」『SQL Anywhere サーバ - プログラミング』

データベースのパーミッションと権限

次に、SQL Anywhere の新しい、または強化されたパーミッションと権限を示します。これらの変更点を使用するには、データベースをアップグレードする必要があります。「SQL Anywhere のアップグレード」392 ページを参照してください。

- **一部の権限の継承サポート** SQL Anywhere で、次のように一部の権限の継承がサポートされるようになりました。
 - PROFILE、READCLIENTFILE、READFILE、WRITECLIENTFILE は継承可能
 - DBA、BACKUP、RESOURCE の各権限は継承不可
 「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。
- **新しい PROFILE 権限** これまでは、ユーザがアプリケーション・プロファイリングや診断トレーシングを実行するには、DBA 権限が必要でした。このリリースでは、PROFILE 権限を持つユーザもこれらの操作を行うことができます。PROFILE 権限の全パーミッションのリストについては、「PROFILE 権限」『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 READFILE 権限** READFILE 権限を使用すると、ユーザは SELECT 文の OPENSTRING 句を使用してファイルから選択できます。「READFILE 権限」『SQL Anywhere サーバ - データベース管理』と「FROM 句」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **新しい READCLIENTFILE 権限** READCLIENTFILE 権限を使用すると、ユーザはクライアント・コンピュータにあるファイルを読み込むことができます。たとえば、ユーザが LOAD TABLE 文を使用して、クライアント・コンピュータにあるファイルからデータをロードする

には、READCLIENTFILE 権限が必要です。「[READCLIENTFILE 権限](#)」『[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 WRITECLIENTFILE 権限** WRITECLIENTFILE 権限を使用すると、ユーザはクライアント・コンピュータにあるファイルに書き込むことができます。たとえば、ユーザが UNLOAD TABLE 文を使用して、クライアント・コンピュータにあるファイルにデータをアンロードするには、WRITECLIENTFILE 権限が必要です。「[WRITECLIENTFILE 権限](#)」『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。
- **DB 領域の CREATE パーミッションのサポート** SQL Anywhere で、指定の DB 領域にデータベース・オブジェクトを作成できる CREATE ON パーミッションがサポートされるようになりました。CREATE パーミッションは、ユーザに割り当てるか、グループから継承できます。「[GRANT 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』と「[パーミッションの概要](#)」『[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・ユーティリティ

次に、SQL Anywhere バージョン 11.0.0 でのデータベース・ユーティリティの強化を示します。

- **設定ファイルの強化** 設定ファイル内でアンパサンド (&) を使用して、前のトークンが次の行に続いていることを指定できるようになりました。「[設定ファイルの使用](#)」『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。
 - **アンロード・ユーティリティ (dbunload) の強化** dbunload が次のように強化されています。
 - 新しいオプション -cp が追加され、dbunload でデータ出力ファイルを圧縮できるようになりました。
 - これまでは、再ロード・オプション -an または -ar を指定しないで暗号化オプション -ek、-ep、または -ea を指定すると、エラーが返されました。このリリースでは、dbunload で暗号化オプションが受け入れられ、作成される出力ファイルに適用されるようになりました。
 - -g オプションによって、MANUAL REFRESH と定義されているテキスト・インデックスが再表示されるようになりました。「[テキスト・インデックス](#)」『[SQL Anywhere サーバ - SQL の使用法](#)』を参照してください。
 - MANUAL REFRESH と定義されているテキスト・インデックスは、デフォルトで再ロード時に初期化されません。これらのテキスト・インデックスを初期化したい場合は、dbunload -g オプションを指定できます。
 - -no オプションを使用して、オブジェクト定義をオブジェクト・タイプ別にアルファベット順でアンロードできます。これはデータベースの *reload.sql* ファイルを比較する場合に便利です。
- 「[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) の強化** これまで、dbvalid ユーティリティでは、デフォルトでテーブルとマテリアライズド・ビューがすべて検証されていました。このリリースでは、dbvalid によって VALIDATE DATABASE 文も実行されます。

検証ユーティリティを実行してデータベースを自動的に起動すると、データベースは読み込み専用モードで起動します。このようにすることで、バックアップとリカバリのプランの一環としてデータベースが検証されている場合にデータベースを変更できないようになっています。

「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。

- **ログ変換ユーティリティ (dbtran) で破損が検出された場合の生成された .sql ファイルの保守** ログ変換ユーティリティの `-k` オプションを使用すると、トランザクション・ログ・ファイルが破損しているためログ・ファイルの変換に失敗した場合に不完全な .sql ファイルを削除しないことを指定できます。「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。

データベース・オプション

次に、SQL Anywhere バージョン 11.0.0 でのデータベース・オプションの強化を示します。

- **allow_read_client_file オプション** このオプションは、クライアント・コンピュータ上のファイルの読み込みを許可するかどうかを制御します。「allow_read_client_file 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。
- **allow_write_client_file オプション** このオプションは、クライアント・コンピュータへのファイルの書き込みを許可するかどうかを制御します。「allow_write_client_file 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。
- **login_procedure オプション** パスワードの有効期限が切れていることを示すエラー・メッセージを通知できるようになりました。「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。
- **max_priority オプション** このオプションは、データベース接続の最高優先度レベルを制御します。「max_priority 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。
- **priority オプション** このオプションは、接続からの要求を実行する優先度レベルを制御します。「priority 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。
- **query_mem_timeout オプション** このオプションは、要求にメモリが付与されるまで待つ時間を制御します。「query_mem_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。

データベース・サーバ・オプション

次に、SQL Anywhere バージョン 11.0.0 でのデータベース・サーバ・オプションの強化を示します。

- **-es サーバ・オプション** SQL Anywhere のこれまでのバージョンでは、`-ec` オプションを指定してデータベース・サーバが起動され(トランスポート・レイヤ・セキュリティのサポートのため)、許可する暗号化プロトコルのリストに NONE または SIMPLE が含まれなかった場合、

共有メモリ・ポートが起動しませんでした。これは、このポートでトランスポート・レイヤ・セキュリティがサポートされていないからです。このため、データベース・サーバへの接続は、強力な暗号化を使用して TCP/IP で行う必要がありました。

-es サーバ・オプションは、共有メモリを経由した、暗号化されていない接続を許可するようにデータベース・サーバに指定します。「[-es 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。

- **-gb サーバ・オプション** サーバ・プロセスの優先度クラスを制御する -gb サーバ・オプションが、Windows に加えて UNIX でサポートされるようになりました。「[-gb 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。
- **-im サーバ・オプション** アプリケーションで、データベースの操作内容がすべて失われてもかまわない場合は、データベースを完全にメモリ内で実行できます。この機能は、データが頻繁に挿入される、高速のテンポラリ・データ・ストアとして SQL Anywhere を使用する場合を対象としています。「[-im 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・コンピュータにあるファイルの読み込みと書き込み** -sf サーバ・オプションを使用して、クライアント・コンピュータにあるファイルの読み込みと書き込みの機能を制御できるようになりました。「[-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。
- **-um サーバ・オプション** -um オプションを使用して、*DBLauncher.app* インスタンスが実行中の場合にこのインスタンスに接続し、*DBLauncher.app* 内の新しいウィンドウにデータベース・サーバ・メッセージを表示することができます。このオプションは Mac OS X だけに適用されます。「[-um 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。
- **Windows パフォーマンス・モニタのオプション** Windows パフォーマンス・モニタをさらに詳細に設定するため、次のサーバ・オプションが追加されました。
 - **-ks オプション** パフォーマンス・モニタで、データベース・サーバからカウンタ値を収集するために使用される共有メモリの作成を無効にします。「[-ks 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。
 - **-ksc オプション** パフォーマンス・モニタでモニタできる接続の最大数を指定します。「[-ksc 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。
 - **-ksd オプション** パフォーマンス・モニタでモニタできるデータベースの最大数を指定します。「[-ksd 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

次に、SQL Anywhere バージョン 11.0.0 でのプロパティとパフォーマンス・モニタの統計の強化を示します。

- **新しい接続プロパティ** このリリースには、次の接続プロパティが追加されています。

- allow_read_client_file
- allow_write_client_file
- AuthType
- CacheReadWorkTable
- ClientNodeAddress
- DiskReadWorkTable
- DiskSyncRead
- DiskSyncWrite
- DiskWaitRead
- DiskWaitWrite
- DiskWriteHint
- DiskWriteHintPages
- LockIndexID
- LockRowID
- max_priority
- OSUser
- priority
- query_mem_timeout
- QueryMemActiveCurr
- QueryMemExtraAvail
- QueryMemGrantFailed
- QueryMemGrantGranted
- QueryMemGrantWaiting
- QueryMemGrantRequested
- QueryMemWaited
- ServerNodeAddress
- ReadHint
- ReadHintScatter

これらのプロパティの説明については、「[接続プロパティ](#)」『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。

- **新しいデータベース・サーバ・プロパティ** このリリースには、次のデータベース・サーバ・プロパティが追加されています。

- DiskRetryRead
- DiskRetryReadScatter
- DiskRetryWrite
- EventTypeDesc
- EventTypeName
- HttpAddresses
- HttpsAddresses
- HttpNumActiveReq
- HttpNumConnections
- HttpNumSessions
- HttpsNumActiveReq
- HttpsNumConnections
- MaxEventType
- MaxRemoteCapability
- MessageCategoryLimit
- OptionWatchAction
- OptionWatchList
- QueryMemActiveCurr
- QueryMemActiveEst
- QueryMemActiveMax
- QueryMemExtraAvail
- QueryMemGrantBase
- QueryMemGrantBaseMI
- QueryMemGrantExtra
- QueryMemGrantFailed
- QueryMemGrantGranted
- QueryMemGrantWaiting
- QueryMemGrantRequested
- QueryMemPages
- QueryMemPercentOfCache
- QueryMemWaited
- ReadHintScatterLimit
- RemoteCapability
- StreamsUsed
- TcpIpAddresses
- WebClientLogFile
- WebClientLogging

これらのプロパティの説明については、「データベース・サーバ・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。

- **新しいデータベース・プロパティ** このリリースには、次のデータベース・プロパティが追加されています。

- AlternateMirrorServerName
- CacheReadWorkTable
- DiskReadWorkTable
- DiskRetryReadScatter
- DiskSyncRead
- DiskSyncWrite
- DiskWaitRead
- DiskWaitWrite
- DiskWriteHint
- DiskWriteHintPages
- HasEndianSwapFix
- MirrorMode
- ReadHint
- ReadHintScatter

これらのプロパティの説明については、「データベース・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** このリリースには、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。
 - キャッシュ読み込み：ワーク・テーブル
 - ディスク読み込み：ワーク・テーブル

システム・プロシージャと関数

次に、SQL Anywhere バージョン 11.0.0 で追加されたシステム・プロシージャと関数の強化を示します。

- **sa_get_dtt_groupreads システム・プロシージャ** 新しい sa_get_dtt_groupreads システム・プロシージャを使用して、データベース・サーバに対してグループ読み込みを発行するコストを推定できます。「sa_get_dtt_groupreads 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。
- **PROPERTY_NAME 関数の強化** 指定した接続レベルで、指定したプロパティ ID を持つプロパティの名前を返すようになりました。「PROPERTY_NAME 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。
- **READ_CLIENT_FILE 関数** 新しい READ_CLIENT_FILE 関数は、クライアント・コンピュータにある、指定したファイルからデータを読み込みます。「READ_CLIENT_FILE 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。
- **WRITE_CLIENT_FILE 関数** 新しい WRITE_CLIENT_FILE 関数は、クライアント・コンピュータにある、指定したファイルにデータを書き込みます。「WRITE_CLIENT_FILE 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。
- **REGEXP_SUBSTR 関数** 新しい REGEXP_SUBSTR 関数を使用すると、文字列内で部分文字列を検索できます。この新しい関数では、引数として正規表現を指定できます。

「[REGEXP_SUBSTR 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_char_terms システム・プロシージャ** 新しい sa_char_terms システム・プロシージャは、CHAR 文字列を単語に分解し、各単語とその位置を返します。「[sa_char_terms 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_nchar_terms システム・プロシージャ** 新しい sa_nchar_terms システム・プロシージャは、NCHAR 文字列を単語に分解し、各単語とその位置を返します。「[sa_nchar_terms 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_text_indexes システム・プロシージャ** 新しい sa_refresh_text_indexes システム・プロシージャは、MANUAL REFRESH または AUTO REFRESH として定義されているすべてのテキスト・インデックスを再表示します。「[sa_refresh_text_indexes 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_text_index_stats システム・プロシージャ** 新しい sa_text_index_stats システム・プロシージャは、データベース内のすべてのテキスト・インデックスに関する統計情報を返します。これには、最終再表示時刻と保留中の変更のサイズが含まれます。「[sa_text_index_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_text_index_vocab システム・プロシージャ** 新しい sa_text_index_vocab システム・プロシージャは、テキスト・インデックスに含まれるすべての単語と、各単語が含まれるインデックス化された値の合計数のリストを返します。「[sa_text_index_vocab 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。

2つの新しいシステム・プロシージャ sa_internal_text_index_vocab と sa_internal_text_index_postings も追加されていますが、これらは sa_text_index_vocab システム・プロシージャでのみ使用されます。

- **sa_text_index_postings システム・プロシージャ** この新しいシステム・プロシージャは内部でのみ使用されます。
- **sa_text_index_handles システム・プロシージャ** この新しいシステム・プロシージャは内部でのみ使用されます。
- **sa_get_user_status システム・プロシージャ** 新しい sa_get_user_status システム・プロシージャを使用すると、ユーザの現在のログイン・ステータスを確認できます。「[sa_get_user_status 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。
- **インボークとしてのプロシージャや関数の実行** プロシージャまたは関数を作成するときに、そのプロシージャまたは関数を、呼び出したユーザ (インボーク) が呼び出したかのように実行するか、それを作成したユーザ (デファイナ) が呼び出したかのように実行するかを指定できます。これを指定するには、CREATE PROCEDURE 文または CREATE FUNCTION 文の SQL SECURITY 句を使用します。「[CREATE FUNCTION 文 \[Web サービス\]](#)」 『[SQL Anywhere サーバ - SQL リファレンス](#)』と「[CREATE PROCEDURE 文 \[Web サービス\]](#)」 『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。

この変更は、外部プロシージャと関数にも適用されます。

- **sa_disk_free_space システム・プロシージャ** sa_disk_free_space システム・プロシージャで、total_space という新しいカラムが返されるようになりました。このカラムは、DB 領域がある

ドライブで使用可能な合計ディスク領域を示します。バージョン 11.0.0 より前の SQL Anywhere で作成されたデータベースでは、データベースをアップグレードするまで total_space カラムは返されません。「[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_external_library_unload システム・プロシージャ** 新しいシステム・プロシージャ sa_external_library_unload が追加され、使用していない外部ライブラリをアンロードできるようになりました。「[sa_external_library_unload 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_index_density システム・プロシージャがスキュー量を返す** sa_index_density システム・プロシージャが強化され、インデックス内にあるスキュー量を返すようになりました。スキューが多いと、バランスがとれているインデックスと比べてパフォーマンスが低くなる可能性があります。「[インデックスの断片化とスキューの削減](#)」『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。
- **sa_materialized_view_info システム・プロシージャの強化** sa_materialized_view_info で返される Status カラムの情報が、Status と DataStatus の 2 つのカラムに分割されました。Status には、ビューが有効であるか、無効であるかに関する情報が返されます。新しい DataStatus カラムには、ビュー内にデータがあるかどうか、またデータが最新かどうかに関する情報が返されます。また、ビューが手動ビューであるか、即時ビューであるかを示す新しいカラム RefreshType が追加されています。「[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_materialized_view_can_be_immediate システム・プロシージャ** 新規に作成するマテリアライズド・ビューはデフォルトで手動ビューになりますが、即時ビューの制約に違反していなければ、即時ビューに変更できます。新しい sa_materialized_view_can_be_immediate システム・プロシージャを使用すると、手動ビューを即時ビューに変更できるかどうかをテストできます。「[sa_materialized_view_can_be_immediate システム・プロシージャ](#)」『SQL Anywhere サーバ - SQL リファレンス』と「[即時ビューの追加の制限](#)」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **sa_post_login_procedure システム・プロシージャ** ユーザのパスワードの有効期限が近いときに警告するかどうかを指定できる新しいシステム・プロシージャが追加されました。「[sa_post_login_procedure 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 関数の強化** EVENT_PARAMETER 関数で、DisconnectReason として abnormal がサポートされるようになりました。この新しい理由は、データベースへの接続を切断する前にクライアント・アプリケーションが異常終了したか、クライアント・コンピュータとサーバ・コンピュータの間で通信エラーが発生した結果、切断が発生したことを示します。「[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。
- **sa_server_option システム・プロシージャの強化** sa_server_option システム・プロシージャに OptionWatchList と OptionWatchAction の 2 つのプロパティが追加されました。これらのプロパティを使用して、データベース・オプションの設定が変更されようとしていないかどうかをモニタし、そのときの処置を指定できます。「[オプション設定のモニタリング](#)」『SQL

[Anywhere サーバ - データベース管理](#)』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。

- **sa_db_properties システム・プロシージャの強化** sa_db_properties システム・プロシージャが、NULL 値を持つ有効なプロパティを返すようになりました。「[sa_db_properties システム・プロシージャ](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。
- **sa_conn_properties システム・プロシージャの強化** sa_conn_properties システム・プロシージャが、NULL 値を持つ有効なプロパティを返すようになりました。「[sa_conn_properties システム・プロシージャ](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。

SQL 文

次に、SQL Anywhere バージョン 11.0.0 での SQL の強化を示します。

- **新しい CALIBRATE GROUP READ 句、ALTER DATABASE 文** ALTER DATABASE 文の新しい CALIBRATE GROUP READ 句を使用して、テンポラリ DB 領域に対してグループ読み込みの調整を実行できます。「[ALTER DATABASE 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。
- **新しい CHECK 句、CREATE MATERIALIZED VIEW 文** CREATE MATERIALIZED VIEW 文の新しい CHECK 句を使用して、ビューの作成前に文を検証できます。「[CREATE MATERIALIZED VIEW 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 RECOMPILE 句、ALTER FUNCTION 文** ALTER FUNCTION 文に新しい句 RECOMPILE が追加され、ユーザ定義関数を再コンパイルできるようになりました。「[ALTER FUNCTION 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 RECOMPILE 句、ALTER PROCEDURE 文** ALTER PROCEDURE 文に新しい句 RECOMPILE が追加され、ストアド・プロシージャを再コンパイルできるようになりました。「[ALTER PROCEDURE 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 REFRESH 句、ALTER MATERIALIZED VIEW 文** ALTER MATERIALIZED VIEW 文に新しい句 REFRESH が追加され、マテリアライズド・ビューのリフレッシュ方法を指定できるようになりました。「[ALTER MATERIALIZED VIEW 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。
- **リカバリとミラーリングをサポートするための LOAD TABLE 文の強化** リカバリとミラーリングをサポートするため、次の句が LOAD TABLE 文に追加されました。
 - **WITH CONTENT LOGGING 句** WITH CONTENT LOGGING 句は、データ・ソースの内容をトランザクション・ログに記録することをデータベース・サーバに指定します。データは、入力が LOAD TABLE によって処理されるときに小さいチャンクで記録されます。これらのチャンクは、ミラーリング・データベースで、またトランザクション・ログを使用したリカバリ時に、ローに再構成できます。WITH CONTENT LOGGING 句は、後でリカバリするために元のデータ・ファイルを維持しないほうがいい場合に便利です。「[LOAD TABLE 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。

- **WITH ROW LOGGING 句** WITH ROW LOGGING 句は、ロードされているすべてのローを、一連の INSERT 文として記録することをデータベース・サーバに指定します。このレベルは、同期するデータベースや、ロード先のテーブルに、計算カラムや CURRENT TIMESTAMP のデフォルトなど、非決定的な値が含まれる場合に最適です。「[LOAD TABLE 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。
- **WITH FILE NAME LOGGING 句** WITH FILE NAME LOGGING 句は、LOAD TABLE 文だけを記録するようにデータベース・サーバに指定します。これがデフォルトの動作であり、SQL Anywhere のこれまでのバージョンでのロギングの動作と同じです。「[LOAD TABLE 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。
- **クライアント・ファイルのロードとアンロードのための新しい句** クライアント・ファイルの新しいロード／アンロード機能をサポートするため、LOAD TABLE 文と UNLOAD TABLE 文が強化されました。
 - **LOAD TABLE 文の新しい USING CLIENT FILE 句** クライアント・コンピュータにあるファイル内のデータを使用してテーブルをロードできます。「[LOAD TABLE 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。
 - **UNLOAD TABLE 文の新しい INTO CLIENT FILE 句** データのアンロード先としてクライアント・コンピュータにあるファイルを指定できます。「[UNLOAD 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。
- **新しいログイン・ポリシー文** 新しいログイン・ポリシー機能をサポートするために、次の文が追加されました。
 - 「[CREATE LOGIN POLICY 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』
 - 「[ALTER LOGIN POLICY 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』
 - 「[DROP LOGIN POLICY 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』
 - 「[CREATE USER 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』
 - 「[ALTER USER 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』
 - 「[DROP USER 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』
- **全文検索用の新しい文と句** 新しい全文検索機能をサポートするために、次の文が追加されました。
 - **新しい CONTAINS 探索条件** CONTAINS 探索条件を使用して、指定するカラムのリスト内に、指定する単語やフレーズがあるかどうかを確認できます。CONTAINS 探索条件は、TRUE または FALSE を返します。複数の単語やフレーズを検索する場合は、さまざまなブール演算子で結合できます。「[CONTAINS 探索条件](#)」『[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 句内の新しい CONTAINS 句** CONTAINS 句は、SELECT 文の FROM 句内で指定します。この句は CONTAINS 探索条件に似ていますが、一致するカラムごとのスコアと、一致する各ローの合計スコアも返します。「[FROM 句](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。
 - **CREATE TEXT CONFIGURATION 文** この文は、テキスト設定オブジェクトを作成します。テキスト設定オブジェクトは、テキスト・インデックスの特性を制御する一連の設定です。「[CREATE TEXT CONFIGURATION 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。

- **ALTER TEXT CONFIGURATION 文** この文は、テキスト設定オブジェクトを変更します。「ALTER TEXT CONFIGURATION 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **DROP TEXT CONFIGURATION 文** この文は、テキスト設定オブジェクトを削除します。「DROP TEXT CONFIGURATION 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **CREATE TEXT INDEX 文** この文は、テキスト・インデックスを作成します。テキスト・インデックスには、インデックス化されたすべてのカラム内の全単語の出現箇所の詳細な位置情報が含まれます。「CREATE TEXT INDEX 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **ALTER TEXT INDEX 文** この文は、テキスト・インデックスを変更します。「ALTER TEXT INDEX 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **DROP TEXT INDEX 文** この文は、データベースからテキスト・インデックスを削除します。「DROP TEXT INDEX 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **REFRESH TEXT INDEX 文** この文は、テキスト・インデックスを再表示します。「REFRESH TEXT INDEX 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **TRUNCATE TEXT INDEX 文** この文は、テキスト・インデックスからデータをトランケートします。「TRUNCATE TEXT INDEX 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **ALTER EVENT 文の強化** ALTER EVENT ... SET HIDDEN 文を使用して、イベント・ハンドラの定義を隠すことができるようになりました。この文を使用すると、ISYSEVENT システム・テーブルの action カラムに格納されているイベント・ハンドラの定義が難読化されます。「ALTER EVENT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **BEGIN SNAPSHOT 文** BEGIN SNAPSHOT 文を使用すると、スナップショット・アイソレーションのためスナップショットをいつ開始するかを制御できます。「BEGIN SNAPSHOT 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **CASE 文と CASE 式の強化** 互換性の向上のため、CASE 文と CASE 式を END または END CASE で終了できるようになりました。「CASE 文」『SQL Anywhere サーバ - SQL リファレンス』と「CASE 式」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **COMMENT 文の強化** ログイン・ポリシー・テーブルと DB 領域にコメントを追加できるようになりました。次の項を参照してください。
 - 「COMMENT 文」『SQL Anywhere サーバ - SQL リファレンス』
 - 「ログイン・ポリシーの管理の概要」『SQL Anywhere サーバ - データベース管理』
 - 「追加 DB 領域の使用」『SQL Anywhere サーバ - データベース管理』
- **CREATE MATERIALIZED VIEW 文の強化** CREATE MATERIALIZED VIEW 文の新しい IMMEDIATE REFRESH 句を使用して、基本となるデータが変更されたときにリフレッシュされるマテリアライズド・ビューを作成できるようになりました。「CREATE MATERIALIZED VIEW 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **DESCRIBE 文の強化** Interactive SQL の DESCRIBE 文を使用して、Interactive SQL が接続しているデータベースまたはデータベース・サーバに関する情報を取得できるようになりました。「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。
 - **IF 文と IF 式の強化** 互換性の向上のため、IF 文と IF 式を ENDIF または END IF で終了できるようになりました。「IF 文」『SQL Anywhere サーバ - SQL リファレンス』と「IF 式」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
 - **LOAD TABLE 文の強化** LOAD TABLE 文の新しい COMPRESSED 句または ENCRYPTED 句を使用して、入力ファイルのデータが圧縮されていることや、暗号化されていることを指定できるようになりました。「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 文の強化**
 - **INDEX 句の強化** INDEX 句を使用してインデックス・ヒントを指定するときに、データベース・サーバで使用する必要があるインデックスを 4 つまで指定できるようになりました。「FROM 句」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
 - **新しい INDEX ONLY 句** INDEX 句を使用してインデックス・ヒントを指定するときに、必要に応じて INDEX ONLY 句を指定して、データベース・サーバでインデックス専用取得 (インデックスのデータのみを使用してクエリを処理する) を行うかどうかを制御できます。「FROM 句」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
 - **新しい CROSS APPLY 句と OUTER APPLY 句** SELECT 文が拡張され、FROM 句で適用式 (CROSS APPLY 句と OUTER APPLY 句) がサポートされるようになりました。適用式は、右側が左側に依存するジョインを簡単に指定できる方法です。たとえば、適用式を使用して、テーブル式内のローごとに 1 回ずつプロシージャまたは派生テーブルを評価できます。「適用式から生成されるジョイン」『SQL Anywhere サーバ - SQL の使用法』と「FROM 句」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
 - **新しい OPENSTRING 句** 新しい OPENSTRING 句を使用すると、SELECT 文を使用してファイル内のデータを問い合わせることができます。「FROM 句」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
 - **イベントの作成時、変更時、削除時、コメント時の所有者の指定** CREATE EVENT、ALTER EVENT、DROP EVENT、COMMENT ON EVENT の各文で、必要に応じて所有者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。
 - 「CREATE EVENT 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』
 - 「DROP EVENT 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』
 - **UNLOAD 文の強化** UNLOAD 文の COMPRESSED 句または ENCRYPTED 句を使用して、アンロード対象のデータの圧縮や暗号化を行うかどうかを指定できるようになりました。「UNLOAD 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- これらの句を使用して圧縮または暗号化したファイルは、LOAD TABLE などを使用して、SQL Anywhere 11.0.0 以降のデータベース・サーバでのみロードできます。その他のツールを使用して圧縮または暗号化したファイルは SQL Anywhere で使用できません。

- **UPDATE 文の強化** 検索と位置付け更新のときに、SET 句を使用してカラム値をデフォルト値に設定できるようになりました。「[UPDATE 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』と「[UPDATE \(位置付け\) 文 \[ESQL\] \[SP\]](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。
- **OPTION 句の拡張** INSERT、UPDATE、DELETE、SELECT、UNION、EXCEPT、INTERSECT の各文の OPTION 句で、user_estimates データベース・オプションの設定を上書きできるようになりました。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 句](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』
 - 「[EXCEPT 句](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』
 - 「[INTERSECT 句](#)」『[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

次に、SQL Anywhere バージョン 11.0.0 でのプログラミング・インタフェースの強化を示します。

- **新しい SQL Anywhere C API** SQL Anywhere C アプリケーション・プログラミング・インタフェース (API) により、PHP、Perl、Python、Ruby など、複数のインタプリタ型プログラミング言語での C や C++ ラッパ・ドライバの作成が簡単になります。SQL Anywhere C API は DBLIB ライブラリの上層に位置し、Embedded SQL で実装されています。

DBLIB に代わるものではありませんが、この API は、C や C++ によるアプリケーションの作成を簡単にします。SQL Anywhere C API を使用するのに、Embedded SQL に関する高度な知識は必要ありません。「[SQL Anywhere C 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。
- **新しい Python データベース API (sqlanydb)** 新しい Python データベース API (sqlanydb) を使用すると、Python で作成されたスクリプトから SQL Anywhere データベースにアクセスできるようになります。sqlanydb モジュールは、Python データベース API 仕様バージョン 2.0 を拡張して実装しています。「[SQL Anywhere Python データベース・サポート](#)」『[SQL Anywhere サーバ - プログラミング](#)』を参照してください。
- **外部環境** SQL Anywhere で、Java、Perl、PHP、CLR、Embedded SQL、ODBC の 6 つの外部ランタイム環境がサポートされるようになりました。これまで SQL Anywhere では、C または C++ で記述されたコンパイル済みネイティブ関数を呼び出すことができました。ただし、これらのプロシージャがサーバで実行されるとき、ダイナミック・リンク・ライブラリまたは共有オブジェクトが常にデータベース・サーバによってロードされ、ネイティブ関数への呼び出しがデータベース・サーバによって行われていました。この方法には、ネイティブ関数が原因で障害が発生した場合、データベース・サーバがクラッシュするというリスクがあります。データベース・サーバの外部環境でコンパイル済みネイティブ関数を実行できると、サーバへのこれらのリスクをなくすることができます。「[SQL Anywhere 外部環境のサポート](#)」『[SQL Anywhere サーバ - プログラミング](#)』を参照してください。

この新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。「[SQL Anywhere のアップグレード](#)」 392 ページを参照してください。

- **PHP 外部環境のサポート** SQL Anywhere 11.0.0 には、5.1.1 ～ 5.1.6 と 5.2.0 ～ 5.2.6 を含むさまざまな PHP バージョン用の構築済みバイナリが含まれます。このいずれかのバージョンをサーバ・コンピュータにインストールしてある場合は、PHP 外部環境を自分で構築しないで SQL Anywhere の構築済みバイナリを使用してください。Linux と Solaris 用には、32 ビットと 64 ビットの両方のバージョンのバイナリが用意されています。Windows とその他のシステム用には、32 ビットのバージョンだけがあります。

上記とは異なる PHP バージョンをインストールしてある場合は、ソフトウェアを構築するか、PHP バージョンを、SQL Anywhere の構築済みバージョンと同じバージョンに切り替える必要があります。SQL Anywhere PHP モジュールの構築方法については、「[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。

- **Perl 外部環境のサポート** Perl 外部環境を使用する前に、SQL Anywhere Perl DBD ドライバのバージョンを更新することが非常に重要です。Perl DBD ドライバを更新しなかった場合、サーバ側 Perl は機能しません。

また、PHP とは異なり、Perl のさまざまなバージョン用の構築済みバイナリは含まれません。SQL Anywhere Perl DBD ドライバのソース・コードは `install-dir¥SDK¥perl` にあります。SQL Anywhere Perl DBD ドライバの構築方法については、「[SQL Anywhere Perl DBD::SQLAnywhere DBI モジュール](#)」 『[SQL Anywhere サーバ - プログラミング](#)』を参照してください。

- **Web サーバでの UTF-8 URL のサポート** これまでは、要求 URL (または要求の本文内の application/x-www-form-urlencoded データ) 内のパーセント記号 (%) でコード化されたデータが、Web サーバによってデータベースの文字セットに復号化されていました。このリリースでは、パーセント記号 (%) でコード化されたデータの内容で、UTF-8 シーケンスの有無がテストされ、最大エクステンツに基づいてデータベースの文字セットに変換されるようになりました。UTF-8 以外でコード化されたデータは、復号化され、すでにデータベースの文字セットになっているかのように処理されます。

クライアント HTTP アプリケーションではパーセント記号 (%) でコード化された UTF-8 データを排他的に送信する必要があります。ASCII は UTF-8 でそのまま表されます。たとえば、スペースは %20 とコード化されます。

- **新しいクライアント・コールバック API** クライアント側でデータのロードとアンロードを実行する新しい機能のサポートのために、新しいクライアント・コールバック API が追加されました。Embedded SQL については、「[db_register_a_callback 関数](#)」 『[SQL Anywhere サーバ - プログラミング](#)』の「DB_CALLBACK_VALIDATE_FILE_TRANSFER」を参照してください。ODBC については、「[SQL Anywhere の接続属性](#)」 『[SQL Anywhere サーバ - プログラミング](#)』の「SA_REGISTER_VALIDATE_FILE_TRANSFER_CALLBACK」を参照してください。

- **SQL_ATTR_CONNECTION_DEAD による切断された接続の検出の高速化** ODBC の SQLGetConnectAttr 呼び出しを使用して SQL_ATTR_CONNECTION_DEAD 属性を取得すると、接続が切断されていた場合、切断後にサーバに要求が送信されていなくても、値 SQL_CD_TRUE が取得されるようになりました。接続が切断したかどうかの確認は、サーバに要求を送信しないで行われ、切断された接続は数秒以内に検出されます。接続が切断されるのには、アイドル・タイムアウトなどの複数の理由があります。この変更の前は、SQL_ATTR_CONNECTION_DEAD で値 SQL_CD_TRUE が取得されたのは、接続が明示的に

切断されたか、接続が切断された後に、SQLExecDirect を呼び出すなどして ODBC ドライバからサーバに要求が送信された場合だけでした。[「接続属性の取得」](#) [『SQL Anywhere サーバ - プログラミング』](#) を参照してください。

- **JDBC ドライバでの ResultSet.getBlob().getBinaryStream() のサポート** iAnywhere JDBC ドライバでは、現在 ResultSet.getBlob() メソッドがサポートされています。このメソッドは、JDBC 仕様ではオプションです。オプションの ResultSet.getBlob().getBinaryStream() メソッドのサポートが追加されました。[「iAnywhere JDBC 3.0 API のサポート」](#) [『SQL Anywhere サーバ - プログラミング』](#) を参照してください。
- **iAnywhere JDBC ドライバで URL ヘッダとして jdbc:odbc に加えて jdbc:ianywhere を許可** これまでは、URL ヘッダ jdbc:odbc を使用するアプリケーションでは、この URL への接続時に JDBC ドライバ・マネージャで iAnywhere JDBC ドライバが使用されると見なすことができました。しかし、Java VM の最近のバージョンでは JDBC ドライバとして Sun JDBC-ODBC ブリッジが登録されるようになり、Sun JDBC-ODBC ブリッジでは jdbc:odbc で始まる URL も許可されるので、アプリケーションに iAnywhere JDBC ドライバではなく Sun JDBC-ODBC ブリッジが使用される可能性が高くなりました。JDBC ドライバ・マネージャで、Sun JDBC-ODBC ブリッジではなく iAnywhere JDBC ドライバが確実に使用されるようにするには、アプリケーションで URL ヘッダ jdbc:ianywhere を使用してください。[「JDBC クライアント・アプリケーションからの接続」](#) [『SQL Anywhere サーバ - プログラミング』](#) を参照してください。
- **ODBC ドライバ・マネージャでの driver=iAnywhere Solutions 11 - Oracle の許可** UNIX ODBC ドライバ・マネージャで driver=iAnywhere Solutions 11 - Oracle が許可されるようになりました。また、アプリケーションがスレッド化されている場合は、スレッド化された Oracle 用 iAnywhere ODBC ドライバがロードされます。アプリケーションがスレッド化されていない場合、ドライバはロードされません。スレッド化されていない Oracle 用 iAnywhere ODBC ドライバはサポートされていません。[「iAnywhere Solutions Oracle ドライバ」](#) [『Mobile Link - サーバ管理』](#) を参照してください。
- **ODBC ドライバ・マネージャでの driver=UltraLite 11 の許可** UNIX ODBC ドライバ・マネージャでは、driver=SQL Anywhere 10 が許可され、SQL Anywhere ODBC ドライバがロードされます (アプリケーションに応じて、スレッド・バージョンまたは非スレッド・バージョン)。UNIX ODBC ドライバ・マネージャでは driver=SQL Anywhere 11 と driver=UltraLite 11 も許可されるようになりました。Ultra Light ドライバの場合、ドライバ・マネージャでは Ultra Light ODBC ドライバのスレッド・バージョンだけがロードされます。このドライバはスレッド・バージョンだけが存在します。
- **TDS 接続の強化** SQL Anywhere データベース・サーバで、Open Client のログイン・サーバ名がデフォルト・データベースの名前と一致しなくてもデフォルト・データベースへの TDS 接続が可能になりました。ただし、接続文字列でデータベースを起動していないこと (DBF=... が無いこと)、またデータベース・サーバでデータベースが 1 つだけ実行されていること、という条件を満たしている必要があります。
- **管理ツールのランチャの再配備の簡素化** データベース・ツール (Sybase Central、DBISQL、DBConsole、ML Monitor) のランチャ実行プログラムの再配備が簡単になりました。JAR ファイルの場所に関するレジストリのエントリやセット・ディレクトリ構造が不要になりました。各実行プログラムには、ツールのロード方法を詳細に示す (実行ファイルと同じ名前の) .ini ファイルが、実行ファイルと同じディレクトリに必要です。[「管理ツールの配備」](#) [『SQL Anywhere サーバ - プログラミング』](#) を参照してください。

- **SQL Anywhere の .NET データ・プロバイダでの分散トランザクションのエンリストのサポート** .NET 2.0 フレームワークで、トランザクション・アプリケーションを記述するためのクラスが含まれる新しいネームスペース System.Transactions が導入されました。クライアント・アプリケーションで 1 つまたは複数の参加者が存在する分散トランザクションを作成し、そのトランザクションに参加できます。クライアント・アプリケーションでは、TransactionScope クラスを使用して、暗黙的にトランザクションを作成できます。接続オブジェクトでは、TransactionScope によって作成されたアンビエント・トランザクションの存在を検出し、自動的にエンリストできます。クライアント・アプリケーションでは、CommittableTransaction を作成し、EnlistTransaction メソッドを呼び出してエンリストすることもできます。

この機能は SQL Anywhere .NET 2.0 データ・プロバイダでサポートされています。分散トランザクションには、大きなパフォーマンスのオーバーヘッドがあります。非分散トランザクションにデータベース・トランザクションを使用することをおすすめします。[「Transaction 処理」](#) [『SQL Anywhere サーバ - プログラミング』](#)を参照してください。

- **SQL Anywhere .NET データ・プロバイダでの名前付きパラメータのサポート** SQL Anywhere プロバイダで、SACommand の名前付きパラメータがサポートされるようになりました。ユーザがすべてのパラメータ名を指定すると、コマンドの実行時にプロバイダによってパラメータ値がマッピングされます。名前付きパラメータを使用するとき、パラメータの順序が、ホスト変数の順序と一致している必要はありません。

```
SACommand cmd = new SACommand(
    "UPDATE MyTable SET name = :name WHERE id = :id", conn );
```

```
SAParameter p1 = new SAParameter(
    "id", SADBType.Integer );
p1.Direction = ParameterDirection.Input;
p1.Value = 1;
cmd.Parameters.Add( p1 );
```

```
SAParameter p2 = new SAParameter(
    "name", SADBType.Char, 40 );
p2.Direction = ParameterDirection.Input;
p2.Value = "asdasd";
cmd.Parameters.Add( p2 );
```

```
cmd.ExecuteNonQuery();
```

- **Web サービスの強化** このリリースでは、次の Web サービスが強化されています。
 - **HTTP:POST タイプの Web クライアント・サービス・プロシージャの拡張によるユーザ定義の本文の許可** CREATE PROCEDURE 文と CREATE FUNCTION 文の TYPE 句が拡張され、MIME タイプを指定できるようになりました。[「CREATE FUNCTION 文 \[Web サービス\]」](#) [『SQL Anywhere サーバ - SQL リファレンス』](#)または[「CREATE PROCEDURE 文 \[Web サービス\]」](#) [『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・クライアント・プロシージャの拡張による PUT、DELETE、HEAD HTTP の各メソッドのサポート** Web サービス・クライアントのプロシージャと関数で PUT、DELETE、HEAD HTTP の各メソッドがサポートされるようになりました。CREATE PROCEDURE 文と CREATE FUNCTION 文の TYPE 句が拡張され、これらのメソッドがサポートされるようになりました。POST メソッドと同様に、PUT では TYPE 句内に Content-Type の拡張を必要とし、1 つの (非代入) パラメータだけが許可されています。[「CREATE SERVICE 文」](#) [『SQL Anywhere サーバ - SQL リファレンス』](#)、[「CREATE FUNCTION 文 \[Web サービス\]」](#) [『SQL Anywhere サーバ - SQL リファレンス』](#)、

「CREATE PROCEDURE 文 [Web サービス]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **sa_http_php_page システム・プロシージャと sa_http_php_page_interpreted システム・プロシージャ** 新しい Web サービス・システム・プロシージャ sa_http_php_page と sa_http_php_page_interpreted は、PHP インタプリタを経由して渡された PHP スクリプトの結果を返します。「sa_http_php_page システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』と「sa_http_php_page_interpreted 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。
- **HTTP_BODY システム関数** Web サービスの新しい関数が追加されました。HTTP_BODY 関数は、HTTP 要求の本文をバイナリ形式で返します。「HTTP_BODY 関数 [HTTP]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **WSDLC での Web サービス・クライアント SOAP プロシージャ生成のサポート** C# と JAVA 用の QAnywhere クライアントサイド SOAP インタフェースの生成に加えて、WSDLC では、SQL Anywhere 用の SQL SOAP (Web サービス) クライアント・プロシージャの生成がサポートされるようになりました。WSDLC では、WSDL1.1 準拠の URL またはファイルが読み込まれ、WSDL 内に記述された対応する各 SOAP 操作にマッピングする、適切なパラメータと句を持つプロシージャ (または関数) が生成されます。生成された SQL 文は SQL ファイルに書き込まれます。「iAnywhere WSDL コンパイラの使用」 『SQL Anywhere サーバ - プログラミング』を参照してください。
- **FORMAT 句を指定して定義した HTTP SOAP サービスに EXPLICIT OFF または ON を追加可能** HTTP SOAP サービスを作成するとき、FORMAT 句のデフォルトは EXPLICIT ON です。このため、DISH サービスによって生成される WSDL では、結果セットで返されるカラムごとに明示的な名前とデータ型が指定されます。したがって、結果セットを表すクライアント側のオブジェクトとインタフェースを SOAP クライアント・ツールキットで自動的に生成でき、カラム値へのネイティブ・アクセスが可能です。この機能が追加される前は、カラム値は、抽象 XML データ要素としてのみアクセスできました。EXPLICIT OFF を指定すると、この以前の動作を有効にできます。

EXPLICIT 応答オブジェクトまたは汎用の SimpleDataset の定義方法の詳細については、「CREATE SERVICE 文」 『SQL Anywhere サーバ - SQL リファレンス』と「チュートリアル: JAX-WS でのデータ型の使用」 『SQL Anywhere サーバ - プログラミング』を参照してください。
- **JSON Web サービスのサポート** SQL Anywhere で、JSON 形式の応答を返す Web サービスがサポートされるようになりました。「CREATE SERVICE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **Web サービスのクライアントのログイン** データベース・サーバで、Web サービス・クライアントの接続を出力ファイルにログインできるようになりました。-zoc サーバ・オプションを指定するか、WebClientLogFile プロパティと WebClientLogging プロパティを sa_server_option システム・プロシージャとともに使用して、ログを制御し、Web サービス・

クライアントのログ・ファイルの場所を指定できます。`-sf` サーバ・オプションを使用してこの機能の使用を無効にすることもできます。次の項を参照してください。

- 「`-zoc` サーバ・オプション」 『SQL Anywhere サーバ - データベース管理』
- 「`-sf` サーバ・オプション」 『SQL Anywhere サーバ - データベース管理』
- 「`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』

Windows Mobile の強化

次に、SQL Anywhere バージョン 11.0.0 での Windows Mobile に関する強化を示します。

- **-gss サーバ・オプションのサポート** Windows CE 4 (Pocket PC 2003) 以降で、`-gss` サーバ・オプションを使用して、内部実行スレッドのデフォルトのスタック・サイズを指定できます。「`-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。
- **チェックサムがデフォルトで有効** データベースが Windows Mobile で実行されているとき、データベースに対してチェックサムが有効になっているかどうかに関係なく、データベース・サーバによってチェックサムが自動的に有効にされます。この機能を使用するには、既存のデータベースをアップグレードするか、データベースを新規に作成する必要があります。「**チェックサムを使用した破損の検出**」 『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

次に、SQL Anywhere バージョン 11.0.0 での UNIX と Linux に関する強化を示します。

- **テンポラリ・ファイルのパーミッションの制御** これまでのリリースでは、データベース・サーバとクライアントによって作成されたテンポラリ・ファイルには、読み込み、書き込み、実行のグローバル・パーミッションがありました。`SATMP` 環境変数を、目的のパーミッションを持つディレクトリに設定することで、テンポラリ・ファイルのパーミッションを制御できます。「`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。
- **SELinux のサポート** SELinux のポリシーを使用して、アプリケーションのシステム・リソースへのアクセスを制御します。Red Hat Enterprise Linux 5 で SQL Anywhere を使用するときデフォルトのポリシーを使用できますが、この方法で実行した場合、SQL Anywhere は保護されません。このリリースでは、Red Hat Enterprise Linux 5 で SQL Anywhere を保護するポリシーが含まれるようになりました。ポリシーを使用するには、コンパイルしてインストールする必要があります。ポリシーのソース・コードが SQL Anywhere のインストールに含まれます。

SQL Anywhere の SELinux ポリシーのコンパイルとインストールについては、`install-dir/selinux/readme` を参照してください。

- **Linux の [アプリケーション] メニュー項目** SQL Anywhere 11 を Linux にインストールするときは、[アプリケーション] メニュー項目を作成することを選択できます。

Mac OS X の強化

- **Mac OS X での暗号化のサポート** Mac OS のデータベース・サーバとクライアントの両方で通信の RSA 暗号化がサポートされるようになりました。強力な暗号化の使用の詳細については、「[トランスポート・レイヤ・セキュリティの概要](#)」『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。
- **Mac OS X での HTTPS のサポート** Mac OS X のデータベース・サーバで HTTPS 通信がサポートされるようになりました。HTTPS の使用の詳細については、「[-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。

その他

次に、SQL Anywhere バージョン 11.0.0 でのその他の強化を示します。

- **無効なビューの変更** これまでは、INVALID ステータスの通常のビューは変更できず、変更が必要な場合はビューを削除してから再作成する必要がありました。このリリースでは、無効なビューの定義を変更して、有効になるようにすることができます。
- **ビッグ・エンディアンとリトル・エンディアンの UTF-16 エンコードのサポート** SQL Anywhere では、プラットフォームのエンディアンに関係なく、すべてのプラットフォームでビッグ・エンディアンとリトル・エンディアンの両方の UTF-16 エンコードがサポートされるようになりました。LOAD TABLE 文と UNLOAD 文、また CSCONVERT 関数で UTF-16 エンコードを使用できます。ただし、接続またはデータベースのエンコードとして UTF-16 エンコードを使用することはできません。「[LOAD TABLE 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』と「[UNLOAD 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。

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。「[SQL Anywhere のアップグレード](#)」 392 ページ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** 特にキャッシュが満杯になったときのインデックスのパフォーマンスが向上しました。インデックスのパフォーマンスを向上するには、インデックスを再構築する必要があります。最も簡単な方法は、データベースを再構築することです。再構築したら、データベース・ファイルが大幅に小さくなっている可能性があります。これは正常な現象であり、問題ではありません。
- **圧縮されたカラムでの INLINE と PREFIX の設定の尊重** これまでは、カラムに対して指定された INLINE と PREFIX の設定は、カラムが圧縮されていた場合は無視され、0 として処理されていました。このリリースでは、カラムが圧縮されていても、カラムに対するこれらの設定が尊重されるようになりました。「[BLOB の格納](#)」『[SQL Anywhere サーバ - SQL の使用法](#)』と「[CREATE TABLE 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。
- **バッチでのホスト変数の使用許可** いくつかの制約付きで、バッチ内でのホスト変数の参照が許可されるようになりました。「[バッチの概要](#)」『[SQL Anywhere サーバ - SQL の使用法](#)』を参照してください。
- **IN リスト・アルゴリズムの強化** これまでは、IN リスト内のすべての要素が定数値であるか、最適化時に評価すると定数値になる場合にのみ、オプティマイザで IN リスト・アルゴリズムが使用されました。このリリースでは、IN リスト述部に、開くときだけに評価される値

(CURRENT DATE、CURRENT TIMESTAMP、非決定的システム関数、ユーザ定義関数) や、クエリ・ブロック 1 回の実行中に定数である値 (外部参照) を含めることができるようになりました。「[InList アルゴリズム \(IN\)](#)」『[SQL Anywhere サーバ - SQL の使用法](#)』を参照してください。

- **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 SELECT 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。
- **新しいデータベースのサイズ縮小** システム・テーブルの次のカラムが圧縮され、新しい (空の) データベースのサイズが約 200 KB 縮小されました。これは、Windows Mobile で使用するデータベースを作成するときに役立ちます。
 - ISYSEVENT.action
 - ISYSJARCOMPONENT.contents
 - ISYSPROCEDURE.proc_defn
 - ISYSSOURCE.source
 - ISYSTEXTCONFIG.char_stoplist
 - ISYSTEXTCONFIG.nchar_stoplist
 - ISYSTRIGGER.trigger_defn
 - ISYSVIEW.view_def
- **デフォルト・パケット・サイズと最小パケット・サイズの拡大** Windows Mobile を除くすべてのオペレーティング・システムで、デフォルトのパケット・サイズが 7300 バイトに拡大されました。Windows Mobile では、デフォルトは 1460 バイトのままです。最小パケット・サイズは 500 バイトに拡大されました。「[CommBufferSize 接続パラメータ \[CBSIZE\]](#)」『[SQL Anywhere サーバ - データベース管理](#)』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。
- **リモート・データ・アクセスのための新しい ODBC クラスのサポート** 次の ODBC クラスのサポートが追加されました。
 - msaccessodbc
 - mysqlodbc
 - ulodbc
 - adsodbc

詳細については、「[ODBC ベースのサーバ・クラス](#)」『[SQL Anywhere サーバ - SQL の使用法](#)』を参照してください。

Access データベースの移行

これまで SQL Anywhere for MS Access 移行ユーティリティ (upsizer ツール) を使用して Microsoft Access データベースを SQL Anywhere に移行していた場合、今後は msaccessodbc クラスを使用できます。

- **データベース・サーバのメッセージの強化** データベース・サーバからのメッセージにカテゴリと重要度が割り当てられるようになりました。この情報には sa_server_messages システム・プロシージャを使用してアクセスできます。また、MessageCategoryLimit プロパティを使用して、維持するメッセージ数を設定できます。「[sa_server_messages 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 列挙体の新しい VALIDATE_COMPLETE パラメータ** a_validate_type 列挙体に、データベースに対して可能な検証をすべて実行できる新しいパラメータ VALIDATE_COMPLETE が追加されました。「[a_validate_db 構造体](#)」『[SQL Anywhere サーバ - プログラミング](#)』を参照してください。
- **外部アンロードの強化** データベースの外部アンロードを行うとき、生成される *reload.sql* の先頭に、コメント付き CREATE DATABASE 文が含まれるようになりました。この文を使用して、アンロードしたデータベースと同じデータベースを作成できます。

アンロードしたデータベースがバージョン 9 以前の SQL Anywhere で作成され、カスタム照合があった場合は、COLLATION 句は次のようになります。

COLLATION collation-label DEFINITION collation-definition

ここで *collation-definition* はカスタム照合を指定する文字列です。

アンロードしたデータベースが強力的な暗号化を使用して作成されていた場合は、CREATE DATABASE 文の KEY 句の値が疑問符 3 つ (???) として表示されます。

詳細については、「[内部アンロードと外部アンロード、内部再ロードと外部再ロード](#)」『[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 Extension Agent OID** このリリースには、次の OID が追加されています。
 - saAgent.saRestart
 - saAgent.saInifile

詳細については、「[SQL Anywhere MIB リファレンス](#)」『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。
- **Deadlock システム・イベント** Deadlock システム・イベントは、デッドロックが発生したときに起動します。イベント・ハンドラでは、sa_report_deadlocks プロシージャを使用して、デッドロックが発生するに至った状況に関する情報を取得できます。Deadlock システム・イベントを使用するには、既存のデータベースをアップグレードする必要があります。「[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。
- **データベースの上限の増加** SQL Anywhere データベースの制限値がいくつか増加しました。「[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。
- **実行プランの変更** オプティマイザによって生成される長いプランに、プラン全体に関連する次のエントリが表示されるようになりました。
 - **見積り済み最良プラン** オプティマイザが検出した最良アクセス・プランの数。
 - **見積り済みプラン** オプティマイザが算出したアクセス・プランの数。
 - **最適化時間** クエリの最適化に要した時間。

「[実行プランの省略形](#)」『[SQL Anywhere サーバ - SQL の使用法](#)』を参照してください。

グラフィカルなプランには次のエントリが表示されるようになりました。

- **見積り済み最良プラン** このエントリはルート・ノードの **[オプティマイザ統計]** セクションにあり、オプティマイザが検出した最良アクセス・プランの数を示します。
- **見積り済みプラン** このエントリはルート・ノードの **[オプティマイザ統計]** セクションにあり、オプティマイザが算出したアクセス・プランの数を示します。
- **最適化時間** このエントリはルート・ノードの **[オプティマイザ統計]** セクションにあり、クエリの最適化に要した時間を示します。
- **FirstRowRunTime** このエントリは **[ノード統計]** セクションにあり、最初のローのフェッチ時間を示します。
- **算出されたジョイン** このエントリはジョイン演算子の **[高度な詳細]** セクションにあり、ジョイン演算子の右側のサブツリーの最適化処理中にオプティマイザが算出したジョイン演算子のリストを示します。
- **事前フィルタ述部** このエントリは **[詳細]** ウィンドウ枠の新しいスキャン・ノードのセクションにあり、スキャンの開始前に評価される述部のリストを示します。
- **スキャン述部** このエントリは **[詳細]** ウィンドウ枠のスキャン・ノードのセクションにあり、ローからフェッチされるカラムとして評価される述部のリストを示します。スキャン述部でローが拒否された場合、それ以上カラムは読み込まれません。スキャン述部は `T.x <= 3` や `T.x IS NULL` などのシンプルな 1 つのカラムの述部です。
- **ポスト・スキャン述部** このエントリは **[詳細]** ウィンドウ枠の新しいスキャン・ノードのセクションにあり、ローがテーブル・ページから読み込まれた直後に評価される述部のリストを示します。ポスト・スキャン述部では、複数のカラムを参照でき、また関数や算術演算を使用できます。
- **未確定述部** このエントリは **[詳細]** ウィンドウ枠の新しいスキャン・ノードのセクションにあり、ローのセットがメモリにフェッチされた後に評価される述部のリストを示します。未確定述部は、通常はサブクエリやユーザ定義関数などの複雑な操作が含まれ、スキャン述部やポスト・スキャン述部として評価できません。
- **算出されたインデックス** **[高度な詳細]** ウィンドウ枠にあるこのエントリは、このスキャン演算子によって参照されているテーブルの最適化処理中にオプティマイザが算出したインデックス・スキャンまたはテーブル・スキャンのリストを示します。リスト内の各項目のフォーマットは、**[詳細]** ウィンドウ枠内のアクセス・プランで使用されているスキャン演算子の詳細と似ています。
- **Primary Key Table** このエントリはインデックス・スキャン演算子の **[インデックス]** セクションにあり、プライマリ・キー・テーブル名を示します。
- **Primary Key Table Estimated Rows** このエントリはインデックス・スキャン演算子の **[インデックス]** セクションにあり、プライマリ・キー・テーブル内のロー数を示します。
- **Primary Key Column** このエントリはインデックス・スキャン演算子の **[インデックス]** セクションにあり、プライマリ・キー・カラムの名前を示します。
- **連続変換** このエントリはインデックス・スキャン演算子の **[インデックス]** セクションにあり、各物理インデックスについて、インデックスのクラスタ化の程度を示す統計を示します。

- **ランダム変換** このエントリはインデックス・スキャン演算子の **[インデックス]** セクションにあり、各物理インデックスについて、インデックスのクラスタ化の程度を示す統計を示します。
- **キー値** このエントリはインデックス・スキャン演算子の **[インデックス]** セクションにあり、インデックス内のユニークなエントリの数を示します。

SQL Anywhere の動作の変更

次に、SQL Anywhere バージョン 11.0.0 での動作の変更を、カテゴリごとに示します。

● **カタログの変更** 次の表は、バージョン 11.0.0 でのカタログの変更点を示しています。

これらの変更点を使用するには、データベースをアップグレードする必要があります。
[「SQL Anywhere のアップグレード」 392 ページ](#)を参照してください。

テーブル名とビュー名	変更点の説明	その他の情報
ISYSTAB/SYSTAB	○ 新しいカラム <code>dbspace_id</code> が追加されました。既存の <code>file_id</code> カラムは最終的にこのカラムによって置き換えられます。 ○ <code>file_id</code> カラムは推奨されなくなりました。今後は <code>dbspace_id</code> を使用してください。グローバル・テンポラリ・テーブルでは、<code>SYSTAB.file_id</code> はシステム DB 領域ではなく、テンポラリ DB 領域を指すようになりました。 ○ テーブルを変更したトランザクションのシーケンス番号を格納する <code>last_modified_tsn</code> という新しいカラムが追加されました。	「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。

テーブル名とビュー名	変更点の説明	その他の情報
ISYSIDX/SYSIDX	○ 新しいカラム dbspace_id が追加されました。既存の file_id カラムは最終的にこのカラムによって置き換えられます。 ○ file_id カラムは推奨されなくなりました。今後は dbspace_id 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 』を参照してください。
ISYSFILE	このシステム・テーブルは推奨されなくなりました。lob_map を除くすべてのカラムは新しい ISYSDBSPACE システム・テーブルにあります。lob_map カラムは新しい ISYSDBFILE システム・テーブルにあります。	次の項を参照してください。 ○ 「SYSDBSpace システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』 ○ 「SYSDBFILE システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』 ○ 「SYSFILE 互換ビュー (旧式)」『SQL Anywhere サーバ - SQL リファレンス』
ISYSDBFILE/SYSDBFILE	DB 領域に関する情報を格納するための新しいテーブルです。	「 SYSDBFILE 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。
ISYSDBSPACE/ SYSDBSpace	DB 領域に関する情報を格納するための新しいテーブルです。	「 SYSDBSpace 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。
SYSDBSpacePERM/ ISYSDBSpacePERM	DB 領域のパーミッションを格納するための新しいテーブルです。	「 SYSDBSpacePERM 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。

テーブル名とビュー名	変更点の説明	その他の情報
ISYSOBJECT/SYSOBJECT	file_id カラムの名前が dbspace_id に変更されました。また、object_type カラムに 17 (テキスト設定) と 18 (DB 領域) の 2 つの新しい値を格納できるようになりました。	「 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。
SYSINDEXES	外部キーとプライマリ・キーのインデックスを他のインデックスと区別するために、indextype フィールドでそれぞれ「外部キー」および「プライマリ・キー」と識別されるようになりました。	「 SYSINDEXES 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。
ISYSCAPABILITYNAME	このテーブルはカタログに含まれなくなりました。対応する SYSCAPABILITYNAME システム・ビューはまだ使用できますが、サーバ・プロパティを使用して生成されます。	「 SYSCAPABILITYNAME 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。
ISYSEVENTTYPE	このテーブルはカタログに含まれなくなりました。対応する SYSEVENTTYPE システム・ビューはまだ使用できますが、サーバ・プロパティを使用して生成されます。	「 SYSEVENTTYPE 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。
ISYSVIEW	マテリアライズド・ビューをリフレッシュしたトランザクションのシーケンス番号を格納する mv_last_refreshed_tsn という新しいカラムが追加されました。	「 SYSVIEW 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。

テーブル名とビュー名	変更点の説明	その他の情報
ISYSLOGINMAP/ SYSLOGINMAP	ログイン・ポリシーに関する情報を格納するための新しいテーブルです。	「 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。
ISYSLOGINPOLICY/ SYSLOGINPOLICY	ログイン・ポリシーに関する情報を格納するための新しいテーブルです。	「 SYSLOGINPOLICY 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。
ISYSLOGINPOLICYOPTION/ SYSLOGINPOLICYOPTION	ログイン・ポリシーに関する情報を格納するための新しいテーブルです。	「 SYSLOGINPOLICYOPTION 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。
ISYSTETEXTCONFIG/ SYSTETEXTCONFIG	テキスト設定オブジェクトに関する情報を格納するための新しいテーブルです。	「 SYSTETEXTCONFIG 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。
ISYSTETEXTIDX/ SYSTETEXTIDX	テキスト・インデックスに関する情報を格納するための新しいテーブルです。	「 SYSTETEXTIDX 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。
ISYSTETEXTIDXTAB/ SYSTETEXTIDXTAB	テキスト・インデックスに関する情報を格納するための新しいテーブルです。	「 SYSTETEXTIDXTAB 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。

- **PHP 関数名の変更** PHP 関数名のプレフィクスは、すべて `sqlanywhere_` から `sasql_` に変更されました。`sqlanywhere_` プレフィクスも関数を呼び出すときにまだ使用できますが、廃止される予定です。新しいプレフィクスを使用するようにアプリケーションを変更してください。
- **INSERT ... ON EXISTING UPDATE 文でのトリガの起動** これまでは、INSERT ... ON EXISTING UPDATE 文を実行したとき、データが更新された場合はトリガは起動しませんでした。このリリースでは、更新に対して文レベルの AFTER トリガが起動します。
- **REFRESH MATERIALIZED VIEW 文** リフレッシュの独立性レベルとして STATEMENT SNAPSHOT と READONLY STATEMENT SNAPSHOT を指定できなくなりました。独立性レベルに SNAPSHOT を指定するとこれらのオプションと同じ効果が得られます。「[REFRESH](#)

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

- **REORGANIZE TABLE 文** 同じテーブルに対して複数の REORGANIZE TABLE 文を同時に実行しようとするエラーが発生します。
- **sa_validate システム・プロシージャ** sa_validate のエクスプレスとチェックサム引数が廃止され、指定しても効果はありません。このリリースでは、デフォルトでチェックサム検証が実行されます。また、引数を指定しないで sa_validate システム・プロシージャを呼び出すと、データベース・サーバでは、すべてのテーブル、マテリアライズド・ビュー、インデックスの検証に加えて、チェックサムを含むデータベース自体が検証されます。「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。
- **-gss サーバ・オプション** Windows XP 以降で -gss サーバ・オプションがサポートされるようになりました。これまでのリリースでは、このオプションは Windows オペレーティング・システムではサポートされていませんでした。「-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。
- **-gx サーバ・オプションのサポート終了** このリリースでは、-gx サーバ・オプションがサポートされなくなりました。SQL Anywhere データベース・サーバの起動時に -gx オプションを指定すると、エラーが発生します。
- **LazyClose 接続パラメータのデフォルト設定が AUTO に変更** これまでのバージョンでは、アプリケーションでカーソルを閉じるとき、LazyClose 接続パラメータが YES に設定されていなかった場合はデータベース・サーバへの往復が必要でした。このリリースでは、カーソルを閉じる要求はデフォルトで複数のカーソル分がキューイングされるようになったので、往復がなくなり、パフォーマンスが向上します。LazyClose 接続パラメータには、YES、NO、AUTO (デフォルト) の3つの値を指定できるようになりました。これまでのリリースでは、YES がデフォルト設定でした。「LazyClose 接続パラメータ [LCLOSE]」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **Embedded SQL のインポート・ライブラリの変更** DBLIB のインポート・ライブラリの Watcom と Borland のバージョンが含まれなくなりました。これらはそれぞれ *dblibtw.lib* と *dblibtb.lib* です。これらのインポート・ライブラリの代わりにインポート定義ファイル (*dblib11.def* ファイル) が用意されています。
- **データベース・ツールのインポート・ライブラリの変更** データベース・ツールのインポート・ライブラリの Watcom と Borland のバージョンが含まれなくなりました。これらはそれぞれ *dbtlstw.lib* と *dbtlstb.lib* です。これらのインポート・ライブラリの代わりにインポート定義ファイル (*dbtool11.def* ファイル) が用意されています。
- **ローを受信しなかったときの DBLIB インジケータの動作の定義** フェッチ時または実行時にデータベース・サーバからローを受信しなかった場合 (エラーが発生したか、結果セットの末尾に到達した場合)、インジケータの値が変更されなくなりました。「インジケータ変数」『SQL Anywhere サーバ - プログラミング』を参照してください。
- **ODBC SQLGetConnectAttr** ODBC の SQLGetConnectAttr 呼び出しを使用して SQL_ATTR_CONNECTION_DEAD 属性を取得すると、接続が切断されていた場合、切断後にサーバに要求が送信されていなくても、値 SQL_CD_TRUE が取得されるようになりました。接続が切断したかどうかの確認は、サーバに要求を送信しないで行われ、切断された接続は

数秒以内に検出されます。接続が切断されるのには、アイドル・タイムアウトなどの複数の理由があります。

これまでのリリースでは、SQL_ATTR_CONNECTION_DEAD で値 SQL_CD_TRUE が取得されたのは、接続が明示的に切断されたか、接続が切断された後に、SQLExecDirect を呼び出すなどして ODBC ドライバからサーバに要求が送信された場合だけでした。

- **utility_db という名前のデータベースを作成または起動できない** utility_db という名前は SQL Anywhere サーバ・ユーティリティ・データベースのために予約されています。utility_db.db という名前のデータベースを新規に作成するか、この名前の既存のデータベースを起動しようとすると、エラーが返されます。utility_db という名前の既存のデータベースがある場合は、別の名前で起動できます。「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。
- **計算カラムの依存性** これまでは、更新操作または挿入操作をエラーなしで続行するには、アプリケーションでトリガを使用して、NOT NULL と宣言されているカラムに NULL 以外の値を割り当てることができました。これは、このカラムに依存する計算カラムに影響し、意図した計算が計算値に反映されない場合があります。このリリースでは、計算カラムが依存する NOT NULL カラムに NULL 値を設定しようとするとエラー・メッセージが表示され、トリガは起動しません。「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。
- **ピリオドを含む DB 領域名のエラー** これまでのリリースでは、引用符で囲んでいない DB 領域名にピリオドが含まれていた場合、DB 領域名のピリオドの前の部分は、サーバで通知されることなく無視されていました。このリリースでは、このような名前があった場合、データベース・サーバでエラーが生成されます。
- **SQL Anywhere Web サーバでの SSL パージョン 2.0 のサポート終了** SQL Anywhere Web サーバを使用するとき、SSL パージョン 3.0 と TLS パージョン 1.0 の接続だけがサポートされます。SSL パージョン 2.0 の接続はサポートされません。
- **CREATE SERVICE オプションの DATATYPE のデフォルト値の変更** DATATYPE 句のデフォルト値が OFF から ON に変更されました。以前の動作が必要な場合は、CREATE SERVICE の定義に DATATYPE OFF を明示的に含めてください。「CREATE SERVICE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **一部の保護された機能の名前変更** このリリースでは、次の保護された機能の名前が変更されています。

これまでの名前	新しい名前
xp_read_file	read_file
xp_write_file	write_file
unload_table	write_file
load_table	read_file

詳細については、「-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。

- **チェックサムの動作の変更** バージョン 11 で作成したデータベースまたはバージョン 11 にアップグレードしたデータベースを、ネットワーク・ドライブやリムーバブル・ドライブなどのメディアで実行した場合、データベース・サーバによってチェックサムが自動的に有効にされます。データベースがこのようなデバイスにあるかぎり、チェックサムは有効なままです。「[チェックサムを使用した破損の検出](#)」『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。
- **HTTP 接続で自動停止されない** これまでのリリースでは、データベースを自動的に停止するように設定すると、HTTP 接続が切断され、データベースへの接続が他になかった場合にデータベースが停止していました。このリリースでは、最後の Command Sequence 接続または TDS 接続が切断したときにのみデータベースが停止します。

データベースへの唯一の接続が HTTP 接続で、自動的に停止するようにデータベースが設定されている場合、HTTP 接続が切断したときにデータベースは自動的に停止しません。また、自動的に停止するように設定されているデータベースに HTTP 接続と Command Sequence 接続または TDS 接続がある場合は、最後の Command Sequence 接続または TDS 接続が切断したときにデータベースが自動的に停止します。このときに HTTP 接続がまだあった場合は切断されます。「[-ga サーバ・オプション](#)」『[SQL Anywhere サーバ - データベース管理](#)』と「[AutoStop 接続パラメータ \[ASTOP\]](#)」『[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・ミラーリングの動作の変更** これまでのリリースでは、プライマリ・サーバまたはミラー・サーバの `-xp` オプションで指定した接続パラメータが無効であった場合、データベース・サーバで接続が繰り返し試行されましたが、接続できませんでした。このリリースでは、`-xp` オプションで指定したパラメータが無効であり、サーバで複数のデータベースが実行されている場合、ミラー・データベースの起動に失敗し、再接続は試行されません。ミラー・データベースが、データベース・サーバで実行されている唯一のデータベースである場合は、データベース・サーバが起動しません。
- **マテリアライズド・ビューのデフォルトのリフレッシュ動作** これまでは、マテリアライズド・ビューのデフォルトのリフレッシュ動作は `WITH EXCLUSIVE MODE` でした。このリリースでは、デフォルトのリフレッシュ動作は、マテリアライズド・ビューが `IMMEDIATE REFRESH` と定義されているかどうか、またデータベースでスナップショット・アイソレーション・レベルが有効になっているかどうかによって異なります。「[REFRESH MATERIALIZED VIEW 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。
- **post_login_procedure データベース・オプションの動作の変更** `post_login_procedure` データベース・オプションのデフォルト設定が `sa_post_login_procedure` システム・プロシージャになりました。「[post_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。
- **non_keywords データベース・オプション** これまでのリリースでは、個々のキーワードの指定に加え、次に示すキーワード・リストの中から特別値を使用して、特定のリリース以降のすべてのキーワードをオフにできました。

`keywords_4_0_d, keywords_4_0_c, keywords_4_0_b, keywords_4_0_a, keywords_4_0,`
`keywords_5_0_01, keywords_5_0`

これらの特別値はサポートされなくなりました。個々のキーワードをオフにすることはできません。「[non_keywords 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。

- **リモート・データ・アクセスでの quoted_identifier データベース・オプション設定の尊重** リモート・データ・アクセスを使用している場合、quoted_identifier オプションのローカル設定を使用して、Adaptive Server Enterprise と Microsoft SQL Server に対する引用符付き識別子の使用を制御できるようになりました。たとえば、quoted_identifiers オプションをローカルで Off に設定すると、Adaptive Server Enterprise に対して引用符付き識別子がオフになります。次の項を参照してください。

○「[サーバ・クラス aseodbc](#)」『[SQL Anywhere サーバ - SQL の使用法](#)』

○「[サーバ・クラス mssodbc](#)」『[SQL Anywhere サーバ - SQL の使用法](#)』

- **precision と scale の各データベース・オプションの範囲の変更** これまでのリリースでは、個々のユーザに precision と scale の各データベース・オプションを設定したり、設定の範囲がテンポラリであることを指定したりできました。ただし、これらの設定はデータベースのリカバリ性に影響する可能性があります。テーブルやドメインを作成または変更する DDL 文の実行時に、テンポラリ設定またはユーザレベルの設定が、対応する PUBLIC 設定と異なる場合、データベースの再構築時に問題が生じる可能性があります。このリリースでは、precision と scale の各データベース・オプションの動作は次のようになっています。

データベース・サーバのバージョン	バージョン 10 以前のデータベース	バージョン 11 のデータベース	バージョン 11 にアップグレードしたデータベース	バージョン 10 以前のデータベースのアンロード
11	PUBLIC 設定は可能 ユーザ設定は可能 テンポラリ設定は 不可	PUBLIC 設定は可能 ユーザ設定は 不可 テンポラリ設定は 不可	PUBLIC 設定は可能 ユーザ設定は 不可 テンポラリ設定は 不可	PUBLIC 設定はアンロードされる ユーザ設定はアンロード時に破棄される
10 以前	PUBLIC 設定は可能 ユーザ設定は可能 テンポラリ設定は可能	なし	なし	PUBLIC 設定はアンロードされる ユーザ設定はアンロードされる

バージョン 10 以前のデータベース・サーバでは、引き続き scale と precision の各オプションをテンポラリに、また個々のユーザに設定できます。

警告

precision と scale の各データベース・オプションのユーザレベル設定またはテンポラリ設定を頼りにしないことをおすすめします。データベースの再構築時に問題が生じる可能性が含まれており、データベース・サーバが予測できない動作をする可能性もあります。

次の項を参照してください。

- 「precision オプション [データベース]」 『SQL Anywhere サーバ - データベース管理』
- 「scale オプション [データベース]」 『SQL Anywhere サーバ - データベース管理』

- **OPTION 句の動作の変更** INSERT、UPDATE、DELETE、SELECT、UNION、EXCEPT、INTERSECT の各文の OPTION 句でサポートされていないデータベース・オプションを指定すると、エラーが返されるようになりました。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 句」 『SQL Anywhere サーバ - SQL リファレンス』
- 「EXCEPT 句」 『SQL Anywhere サーバ - SQL リファレンス』
- 「INTERSECT 句」 『SQL Anywhere サーバ - SQL リファレンス』

- **読み込み専用データベースのロールバック・ログの動作の変更** これまでのリリースでは、トランザクション志向のテンポラリ・オブジェクトがある読み込み専用データベースの操作はトランザクション志向として処理されず、ロールバック・ログ情報が維持されませんでした。このリリースでは、読み込み専用データベース内のトランザクション志向テンポラリ・オブジェクトに完全にトランザクション志向のセマンティックがあります。これらのオブジェクトは、コミット、ロールバック、セーブポイントまでのロールバックの対象となります。

- **Itanium 64 ビットのサポート対象プラットフォームの変更** これまでのバージョンでは、Itanium II チップ搭載の Windows Server 2003 を対象としたソフトウェアの 64 ビット通常版と、64 ビットの Linux と HP-UX の各オペレーティング・システムを対象としたクライアント／サーバ限定版がありました。

このリリースでは、64 ビット HP-UX を対象としたクライアント／サーバ限定版だけを使用できます。

- **アンロード・ユーティリティ (dbunload) の動作の変更** これまでのリリースでは、dbunload の -ea、-ck、-ep の各オプションを、-an オプションまたは -ar オプションと同時に指定して、新しいデータベースの暗号化を制御する必要がありました。このリリースでは、データベースの全体またはその一部をアンロードした後に再ロードしなかった場合、-ea、-ck、-ep の各オプションによって、作成されるテーブル・データ・ファイルの暗号化が制御されます。これらのファイルを使用して Interactive SQL からデータベースを再ロードするとき、READ 文のパラメータとして暗号化キーを指定する必要があります。「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。

また、これまでのリリースでは、データベースの抽出に使用する dbunload のバージョンが、データベースを実行しているデータベース・サーバと同じバージョンである必要はありませんでした。このリリースでは、バージョン 10.0.0 以降のデータベースで dbunload を使用する場合、使用する dbunload のバージョンが、データベースへのアクセスに使用するデータベー

ス・サーバのバージョンと一致する必要があります。dbunload のバージョンがデータベース・サーバのバージョンより古いまたは新しい場合は、エラーがレポートされます。

- **抽出ユーティリティ (dbxtract) の動作の変更** これまでのリリースでは、データベースの抽出に使用する dbxtract のバージョンが、データベースを実行しているデータベース・サーバと同じバージョンである必要はありませんでした。このリリースでは、バージョン 10.0.0 以降のデータベースで dbxtract を使用する場合、使用する dbxtract のバージョンが、データベースへのアクセスに使用するデータベース・サーバのバージョンと一致する必要があります。dbxtract のバージョンがデータベース・サーバのバージョンより古いまたは新しい場合は、エラーがレポートされます。
- **ロックの動作の変更** これまでのリリースでは、独立性レベル 0 で実行されている UPDATE 文または DELETE 文が、文の処理対象ではないローのロー・ロックをブロックする場合があります。このリリースでは、UPDATE 文または DELETE 文が、文の処理対象ではないローに意図的ロックまたは排他ロックをかける可能性は低くなっています。アプリケーションを開発する際は、UPDATE 文と DELETE 文に独立性レベル 0 または 1 を使用する場合は注意し、動作がアプリケーションで許容可能であることを確認する必要があります。[「更新時のロック」『SQL Anywhere サーバ - SQL の使用法』](#)と[「削除時のロック」『SQL Anywhere サーバ - SQL の使用法』](#)を参照してください。
- **プロパティ名の変更** このリリースでは、次のプロパティの名前が変更されています。

以前の名前	新しい名前
CacheHitsEng	CacheHits
CacheReadEng	CacheRead
DiskReadEng	DiskRead
ReadHint	DiskReadHint
ReadHintScatter	DiskReadHintPages
ReadHintScatterLimit	DiskReadHintScatterLimit

詳細については、[「接続、データベース、データベース・サーバのプロパティ」『SQL Anywhere サーバ - データベース管理』](#)を参照してください。

- **言語選択ユーティリティ (dblang)** これまでのリリースでは、このユーティリティは、インストール時に多言語リソース開発キット (IRDK) を選択した場合にのみインストールされていました。このリリースでは、すべての多言語リソースと言語選択ユーティリティ (dblang) が常にインストールされます。
- **テンポラリ・テーブルとインデックスのデフォルトの DB 領域** テンポラリ・テーブルは TEMPORARY DB 領域だけに作成できます。CREATE TABLE 文の IN 句で SYSTEM DB 領域を指定すると、IN 句は無視され、テンポラリ・テーブルはテンポラリ DB 領域に作成されます。CREATE TABLE 文の IN 句でユーザ定義の DB 領域を指定すると、エラーが返されます。また、テンポラリ・オブジェクト作成時は default_dbpace オプションが無視されます。

- **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とき、ON COMMIT DELETE と指定されたローカル・テンポラリ・テーブルをロードできなくなりました。これまでのリリースでは、ON COMMIT DELETE ROWS と指定して定義されたローカル・テンポラリ・テーブルにデータをロードできました。

このリリースでは、LOAD TABLE 文を実行すると、オートコミットが自動的に実行されます。これまでのリリースではオートコミットは実行されない場合もありました。

- **データベース・サーバ・オプション** サーバ・オプション `-uc` と `-ui` が Mac OS X でサポートされるようになりました。これまでは、Linux だけでサポートされていました。Linux では、`-ui` サーバ・オプションを使用すると、[サーバ起動オプション] ウィンドウが開き、データベース・サーバ・メッセージ・ウィンドウが表示され、X-Window Server が起動するかどうかにかかわらず、データベース・サーバが起動します。Mac OS X では、`-ui` を使用すると、データベース・サーバ・メッセージが新しいウィンドウに表示され、使用可能な表示がない場合はデータベース・サーバがシェル・モードで起動します。`-uc` サーバ・オプションでは、データベース・サーバがシェル・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』と「[-um 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。

- **Unicode 呼び出しをサポートしない ODBC ドライバでリモート・データ・アクセスが機能しない** Unicode 呼び出しをサポートしない ODBC ドライバでリモート・データ・アクセスが機能なくなりました。このため、Unicode 以外の ODBC ドライバの場合、リモート・データ・アクセスでは ODBC ドライバからのデータに対して文字セットの変換が行われません。

- **SYSFILE システム・ビュー** SYSFILE 互換ビューにテンポラリ・ファイル用のローが含めるようになりました。「[SYSFILE 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。

SQL Anywhere の廃止予定機能とサポート終了機能

- **LOAD TABLE、UNLOAD TABLE、INPUT、OUTPUT の各文の FORMAT ASCII 句は廃止予定** LOAD TABLE、UNLOAD TABLE、INPUT、OUTPUT の各文の FORMAT ASCII 句が推奨されなくなり、FORMAT TEXT に置き換えられました。dbunload などのユーティリティでは、FORMAT ASCII ではなく FORMAT TEXT を含む再ロード・スクリプトが生成されるようになりました。

OUTPUT 文では、FORMAT TEXT 句によって、旧バージョンでの FORMAT ASCII と同じファイル形式でデータが書き込まれます。これまで FORMAT TEXT によって作成されていた出力は使用できなくなりました。

- **データベース・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。
 - MapPages
 - PreserveSource
 - UniqueIdentifier

- **サーバ・プロパティ** このリリースでは、次のサーバ・プロパティが推奨されなくなっています。

- MaxMessage
- Message
- MessageTime
- MessageText
- MessageWindowSize

- **SPX プロトコルのサポート終了** このリリースでは、SPX プロトコルがサポートされなくなりました。その結果、次のプロトコル・オプションがサポートされなくなりました。

- ExtendedName プロトコル・オプション [ENAME]
- RegisterBindery プロトコル・オプション [REGBIN]
- SearchBindery プロトコル・オプション [BINSEARCH]

SQL Anywhere .NET データ・プロバイダの次の機能がサポートされなくなりました。

- SACommLinksOptionsBuilder クラス : SpxOptionsBuilder プロパティ
- SACommLinksOptionsBuilder クラス : SpxOptionsString プロパティ
- SASpxOptionsBuilder クラス

- **dbinit -e オプションのサポート終了** 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 オプションはサポートされなくなりました。単純暗号化を指定するには -ea simple オプション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。

- **サポート終了したデータベース・オプション** このリリースでは、次のデータベース・オプションと、対応するデータベース・プロパティがサポートされなくなりました。

オプション	このリリースでの動作
ansi_integer_overflow	オーバフローが発生すると必ず SQLSTATE = 22003 - overflow エラーが発生するようになりました。 マテリアライズド・ビューがある古いデータベースにアンロードまたは接続するときは、このオプションの設定は無視されます。
ansi_substring	SUBSTRING 関数の動作が ANSI/ISO SQL/2003 の動作と一致するようになりました。「 SUBSTRING 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。
automatic_timestamp	TIMESTAMP データ型の新規カラムに明示的なデフォルト値が定義されていない場合、デフォルト値として Transact-SQL の timestamp が適用されなくなりました。

オプション	このリリースでの動作
divide_by_zero_error	ゼロによる除算があると、SQLSTATE 22012 のエラーが発生します。 マテリアライズド・ビューがある古いデータベースにアンロードまたは接続するときは、このオプションの設定は無視されます。
float_as_double	SQL Anywhere で、精度が指定されていない場合、FLOAT キーワードが Adaptive Server Enterprise の FLOAT キーワードと同じように動作しません。SQL Anywhere では、FLOAT 値は DOUBLE 値と同じように処理されません。 Open Client 接続と jConnect 接続では、この動作は、これまでのリリースのデフォルトの動作と異なります。 マテリアライズド・ビューがある古いデータベースにアンロードまたは接続するときは、このオプションの設定は無視されます。
optimistic_wait_for_commit	このオプションはサポートされなくなりました。
query_plan_on_open	カーソルに対して OPEN を実行したときにプランが返されなくなりました。EXPLAIN 文か PLAN 機能を使用すると詳細な説明が得られます。 「EXPLAIN 文 [ESQL]」 『SQL Anywhere サーバ - SQL リファレンス』 と 「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。
ri_trigger_time	UPDATE または DELETE の後に参照整合性アクションが実行されるようになりました。
truncate_with_auto_commit	TRUNCATE TABLE 文の実行前と実行後の両方に COMMIT が実行されるようになりました。
tsql_hex_constant	16 進定数がバイナリ文字定数として処理されるようになりました。
uuid_has_hyphens	UUID 文字列に 4 つのハイフンが含まれるようになりました。 マテリアライズド・ビューがある古いデータベースにアンロードまたは接続するときは、このオプションの設定は無視されます。

オプション	このリリースでの動作
percent_as_comment	これまでのリリースでは、percent_as_comment データベース・オプションの設定によっては、パーセント記号 (%) をコメント・マーカとして使用できました。このリリースでは、% 記号は SQL Anywhere でモジュロ演算子として処理されるようになりました。「 MOD 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。

- **SQLANY10 環境変数のサポート終了** これまでのリリースでは、SQL Anywhere ソフトウェアの一部が共有ディレクトリにインストールされていました。この場所は SQLANYSH10 環境変数で指定できました。このリリースでは、インストール・プロセスによってソフトウェアが共有ディレクトリにインストールされなくなり、SQLANYSH10 環境変数は使用されなくなりました。

サイレント・インストールを作成するときに SHARED_DIR の場所を設定する必要がなくなりました。「[サイレント・インストールを使用した配備](#)」『[SQL Anywhere サーバ - プログラミング](#)』を参照してください。

- **sa_get_server_messages システム・プロシージャのサポート終了** これまでのリリースでは、sa_get_server_messages システム・プロシージャを使用して、データベース・サーバ・メッセージ・ウィンドウから結果セットとして定数を返すことができました。このリリースでは、sa_server_messages システム・プロシージャを使用して同じ情報を取得できるようになりました。「[sa_server_messages 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。

- **background_priority オプションは廃止予定** background_priority オプションは推奨されなくなりました。今後は priority オプションを使用してください。「[priority 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。

- **encrypt_aes_random_iv オプションのサポート終了** このリリースでは、encrypt_aes_random_iv データベース・オプションがサポートされなくなりました。ランダム IV (初期化ベクトル) が常に使用されるようになりました。

- **DLL プロトコル・オプションのサポート終了** DLL プロトコル・オプションがサポートされなくなりました。Windows データベース・サーバとクライアントでは Winsock 2.2 が使用されます。Windows Mobile クライアントでは Winsock 1.1 が使用されます。「[TCP/IP プロトコルの使用](#)」『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。

- **SQL Anywhere Broadcast Repeater ユーティリティの名前変更** バージョン 10 では、SQL Anywhere Broadcast Repeater ユーティリティを実行するコマンドは dbns10 でした。このリリースでは、dbns11 です。「[Broadcast Repeater ユーティリティ \(dbns11\)](#)」『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。

- **SQLANY10 と SQLANYSAMP10 の各環境変数の名前変更** SQLANY10 と SQLANYSAMP10 の各環境変数は、それぞれ SQLANY11 と SQLANYSAMP11 に名前が変更されました。次の項を参照してください。

- 「[SQLANY11 環境変数](#)」『[SQL Anywhere サーバ - データベース管理](#)』
- 「[SQLANYSAMP11 環境変数](#)」『[SQL Anywhere サーバ - データベース管理](#)』

Mobile Link

次の項では、Mobile Link バージョン 11.0.0 の新機能、動作の変更、廃止予定機能について説明します。

Mobile Link の新機能

次に、バージョン 11.0.0 で導入された Mobile Link の追加機能を示します。

統合データベース

- **DB2 メインフレームを統合データベースとしてサポート** Mobile Link では、これまで DB2 LUW (Linux、UNIX、Windows) が統合データベースとしてサポートされていました。このリリースでは、DB2 メインフレームもサポートされるようになりました。
「[IBM DB2 メインフレーム統合データベース](#)」 『[Mobile Link - サーバ管理](#)』を参照してください。
- **MySQL を統合データベースとしてサポート** Mobile Link で、統合データベースとして MySQL がサポートされるようになりました。
「[MySQL 統合データベース](#)」 『[Mobile Link - サーバ管理](#)』を参照してください。
- **Mobile Link システム・データベース (MLSD)** 個別データベースに Mobile Link システム・データ (MLSD - Mobile Link システム・データベース) を保持できるようになりました。この機能は Microsoft DTC (分散トランザクション・コーディネータ) とともに使用してください。
「[-cs オプション](#)」 『[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

- **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_qa_delivery_archive](#)」 『[Mobile Link - サーバ管理](#)』
 - 「[ml_qa_repository_archive](#)」 『[Mobile Link - サーバ管理](#)』
 - 「[ml_qa_repository_props_archive](#)」 『[Mobile Link - サーバ管理](#)』
 - 「[ml_qa_status_history_archive](#)」 『[Mobile Link - サーバ管理](#)』
 - 「[ml_server](#)」 『[Mobile Link - サーバ管理](#)』
 - 「[ml_active_remote_id](#)」 『[Mobile Link - サーバ管理](#)』
 - 「[ml_passthrough](#)」 『[Mobile Link - サーバ管理](#)』
 - 「[ml_passthrough_repair](#)」 『[Mobile Link - サーバ管理](#)』
 - 「[ml_passthrough_script](#)」 『[Mobile Link - サーバ管理](#)』
 - 「[ml_passthrough_status](#)」 『[Mobile Link - サーバ管理](#)』

- 複数の新しい Mobile Link システム・プロシージャが追加されました。次の項を参照してください。

- 「[ml_add_passthrough システム・プロシージャ](#)」 『Mobile Link - サーバ管理』
- 「[ml_add_passthrough_repair システム・プロシージャ](#)」 『Mobile Link - サーバ管理』
- 「[ml_add_passthrough_script システム・プロシージャ](#)」 『Mobile Link - サーバ管理』
- 「[ml_delete_passthrough システム・プロシージャ](#)」 『Mobile Link - サーバ管理』
- 「[ml_delete_passthrough_repair システム・プロシージャ](#)」 『Mobile Link - サーバ管理』
- 「[ml_delete_passthrough_script システム・プロシージャ](#)」 『Mobile Link - サーバ管理』

iAnywhere Solutions Oracle ドライバ

- **Oracle DSN が暗号化されたパスワードを格納** Windows ODBC アドミニストレータで Oracle ODBC DSN を作成するときに、DSN に格納されたパスワードを暗号化するように選択できるようになりました。「[iAnywhere Solutions Oracle ドライバ](#)」 『Mobile Link - サーバ管理』 を参照してください。
- **Oracle ODBC ドライバが Microsoft 分散トランザクションをサポート** Oracle ODBC ドライバで Microsoft 分散トランザクションがサポートされるようになりました。Windows ODBC アドミニストレータで、[Microsoft 分散トランザクションを有効にする] を選択し、Oracle クライアントとともに適切な DLL がインストールされていることを確認してください。「[iAnywhere Solutions Oracle ドライバ](#)」 『Mobile Link - サーバ管理』 を参照してください。

Mobile Link サーバ

- **リレー・サーバ** リレー・サーバは、Mobile Link サーバ、Afaria サーバ、および OneBridge サーバと、モバイル・デバイスとの間の、安全で負荷分散された Web サーバ経由の通信を実現する、Web 拡張機能セットです。「[リレー・サーバの概要](#)」 『Mobile Link - サーバ管理』 を参照してください。
- **Sybase リレー・サーバのホスティング・サービス** Sybase リレー・サーバのホスティング・サービスは、Sybase をホストとするリレー・サーバのファームです。Mobile Link データ同期を使用するモバイル・アプリケーションの開発を容易にすること、また特に公共無線ネットワークを使用してデータを送信する場合に開発者による評価プロセスを簡素化することを目的としています。「[Sybase リレー・サーバのホスティング・サービス](#)」 『Mobile Link - サーバ管理』 を参照してください。
- **Mobile Link サーバ・ファーム** Mobile Link サーバを同一のサーバ・ファームに明示的にグループ化できるようになりました。

同一のリモート ID による冗長な同時同期をファーム全体から自動的に検出できるようになりました。これにより、負荷分散によって同一のリモート ID を Mobile Link サーバに送信し続ける必要がなくなりました。

各 Mobile Link サーバに対して同一の設定ができるようになります。

Notifier と QAnywhere コネクタに対するフェールオーバーが自動的にサポートされています。ファームにより、自動的に Mobile Link サーバが指定されて Notifier と QAnywhere コネクタが

実行されます。最初に指定されたコンピュータでフェールした場合は、新しいサーバが選択されて Notifier と QAnywhere コネクタが実行されます。

- **64 ビットのプラットフォーム** いくつかの 64 ビット・プラットフォームで Mobile Link サーバが完全に 64 ビット対応になりました。サポートされる 64 ビット・プラットフォームのリストについては、http://www.iAnywhere.jp/developers/technotes/os_components_1101.html を参照してください。
- **Java DownloadTableData インタフェースの新しいメンバ** `getLastDownloadTime` メソッドによってテーブルの最終ダウンロード時刻が返されます。「[getLastDownloadTime メソッド](#)」『[Mobile Link - サーバ管理](#)』を参照してください。
- **SQL パススルー** SQL パススルー機能によって、統合データベースから SQL Anywhere または Ultra Light のクライアントに SQL 文のスクリプトをダウンロードし、クライアント上で適切な時間に SQL 文を実行させることができます。「[SQL パススルーの概要](#)」『[Mobile Link - クライアント管理](#)』を参照してください。
- **情報メッセージの待機** 情報の行がログに出力されたときに通知を受け取るように登録するための Java と .NET API のメソッドが追加されました。

次の項を参照してください。

- 「INFO 変数」『[Mobile Link - サーバ管理](#)』
- 「`addInfoListener` メソッド」『[Mobile Link - サーバ管理](#)』
- 「`removeInfoListener` メソッド」『[Mobile Link - サーバ管理](#)』
- 「INFO フィールド」『[Mobile Link - サーバ管理](#)』
- 「`LogCallback InfoListener` イベント」『[Mobile Link - サーバ管理](#)』

mlsrv11 の新機能

- **-cs オプション** システム・テーブル、プロシージャ、トリガ、ビューなどの Mobile Link サーバ・システム・オブジェクトを、統合データベース以外のデータベースに格納できるようになりました。Mobile Link システム オブジェクトを格納するデータベースは、Mobile Link システム・データベース (MLSD) と呼ばれます。

MLSD の接続パラメータを指定するには、`-cs` を使用します。「[-cs オプション](#)」『[Mobile Link - サーバ管理](#)』を参照してください。

- **-lsc オプション** ローカル・サーバの接続情報を指定します。この情報は、サーバ・ファームのその他のサーバに渡されます。「[-lsc オプション](#)」『[Mobile Link - サーバ管理](#)』を参照してください。
- **-ss オプション** Mobile Link サーバをサーバ・ファームで実行できます。「[-ss オプション](#)」『[Mobile Link - サーバ管理](#)』を参照してください。

別途ライセンスが必要な必須コンポーネント

`-ss` オプションは、Mobile Link の高可用性オプションの機能であり、別途ライセンスが必要です。「[別途ライセンスが必要なコンポーネント](#)」『[SQL Anywhere 11 - 紹介](#)』を参照してください。

- **-tc オプション** SQL スクリプト実行のカウント・ダウン・タイマを設定します。「[-tc オプション](#)」『[Mobile Link - サーバ管理](#)』を参照してください。

- **-tf オプション** -tc を使用して指定されたカウント・ダウン・タイマの期限が切れたら、SQL スクリプトを実行できないようにします (Oracle は対象外)。「[-tf オプション](#)」『[Mobile Link - サーバ管理](#)』を参照してください。

Mobile Link の新しいスクリプト機能

- **非ブロッキング・ダウンロード ACK スクリプト** 「[nonblocking_download_ack 接続イベント](#)」『[Mobile Link - サーバ管理](#)』と「[publication_nonblocking_download_ack 接続イベント](#)」『[Mobile Link - サーバ管理](#)』を参照してください。

Mobile Link リダイレクタの強化

- **リダイレクタは廃止予定** リダイレクタは推奨されなくなり、リレー・サーバに置き換えられました。「[リレー・サーバ](#)」『[Mobile Link - サーバ管理](#)』を参照してください。
- **新しいリレー・サーバ** リレー・サーバは、Mobile Link サーバ、Afaria サーバ、および OneBridge サーバと、モバイル・デバイスとの間の、安全で負荷分散された Web サーバ経由の通信を実現する、Web 拡張機能セットです。「[リレー・サーバの概要](#)」『[Mobile Link - サーバ管理](#)』を参照してください。

Mobile Link クライアント

SQL Anywhere クライアント

- **C++ および .NET 用の dbmlsync API** dbmlsync API は、C++ または .NET で記述された Mobile Link クライアント・アプリケーションが同期を起動し、要求した同期の進行状況に関するフィードバックを受け取れるようにするプログラミング・インタフェースです。「[dbmlsync API](#)」『[Mobile Link - クライアント管理](#)』を参照してください。
- **同期プロファイル** dbmlsync コマンド・ラインを同期プロファイルとしてデータベースに格納できるようになりました。dbmlsync の -sp オプションを使用すると、同期プロファイルからコマンド・ライン同期オプションにオプションを追加できます。「[-sp オプション](#)」『[Mobile Link - クライアント管理](#)』を参照してください。
- **挿入操作でロードされたローをロギングする LOAD TABLE のオプション機能** 新しい WITH ROW LOGGING 句によって、SQL Anywhere リモート・データベースで LOAD TABLE 文を使用してデータをロードできるようになりました。これまでは、LOAD TABLE ではローのログが取られずに dbmlsync から無視されていたため、LOAD TABLE は同期環境において必ずしも有益ではありませんでした。「[LOAD TABLE 文を使用したデータのインポート](#)」『[SQL Anywhere サーバ - SQL の使用法](#)』を参照してください。
- **dbmlsync ログ・スキンの最適化** 重複のないパブリケーション用に最適化されています。

セキュリティ

このリリースでは、次のセキュリティ機能が追加されています。

- **エンドツーエンド暗号化** Mobile Link 同期ストリームおよびクライアントで、プロトコルレベルでエンドツーエンド暗号化がサポートされるようになりました。「[エンドツーエンド暗号化](#)」『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。
- **キー・ペア・ジェネレータ・ユーティリティ (createkey)** このユーティリティでは、Mobile Link エンドツーエンド暗号化で使用する RSA と ECC のキー・ペアを作成します。「[キー・ペア・ジェネレータ・ユーティリティ \(createkey\)](#)」『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。

サーバ起動同期

- **Mobile Link サーバ・ファームのサポート** SIS が強化され、Mobile Link サーバ・ファーム環境での動作が改善されました。ファーム内のすべての Mobile Link サーバで Notifier を実行できるようになりました。これらの Notifier によって、同じ Listener への冗長な通知がなくなります。「[-lsc オプション](#)」『[Mobile Link - サーバ管理](#)』を参照してください。

Listener の強化

- **新しい dblsn オプション** Windows 用の Mobile Link Listener に次のオプションが追加されています。
 - **-ni** IP 追跡を無効にします。
 - **-pc-** 永続的な接続を無効にします。
 - **-nu** デフォルトの UDP 受信を無効にします。

Mobile Link の動作の変更と廃止予定機能

次に、バージョン 11.0.0 で導入された Mobile Link の変更点を示します。

Mobile Link サーバの変更

- **デフォルトのダウンロード確認の変更** -nba オプションのデフォルト値が -nba+ になり、デフォルトで非ブロッキング・ダウンロード確認が実行されるようになりました。既存の配備環境で、ダウンロード確認をブロックするダウンロードを使用する場合、次のいずれかの操作を行ってください。
 - **-nba-** オプションを使用する。
 - 同期スクリプトを確認して nonblocking_download_ack スクリプトか publication_nonblocking_download_ack スクリプトまたはその両方を使用する (推奨)。
- **certificate と certificate_password の各プロトコル・オプションの名前変更** TLS および HTTPS の certificate と certificate_password の各プロトコル・オプションは、それぞれ identity と identity_password に名前が変更されました。Mobile Link サーバの「[-x オプション](#)」『[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

- **dbmlsync 統合コンポーネントは廃止予定** dbmlsync 統合コンポーネントは推奨されなくなり、dbmlsync API に置き換えられました。
「[dbmlsync API](#)」 [『Mobile Link - クライアント管理』](#) を参照してください。
- **dbmlsync StreamCompression 拡張オプションのサポート終了** このオプションはサポートされなくなりました。
- **-lt 拡張オプションのデフォルトは OFF に変更** dbmlsync の LockTables (-lt) 拡張オプションのデフォルトは以前は ON でした。このリリースでは、デフォルトが OFF になり、デフォルトでは dbmlsync によって同期テーブルがロックされません。「[LockTables \(lt\) 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。

その他の Mobile Link の動作の変更

サーバ起動同期

- **Sierra Wireless Aircards のサポート終了** Sierra Wireless Aircards の SMS 受信ライブラリは、サポートされなくなりました。
- **-g オプションは廃止予定** -g Listener オプションは -ni オプションに置き換えられました。
- **-ga オプションは廃止予定** -ga Listener オプションは推奨されなくなりました。非同期の IP 追跡は、暗黙的に行われます。
- **-gi デフォルトの変更** デフォルトのポーリング間隔が、10 秒から 60 秒に変更されました。

QAnywhere

次の項では、QAnywhere バージョン 11.0.0 の新機能、動作の変更、廃止予定機能について説明します。

QAnywhere の新機能

次に、バージョン 11.0.0 で導入された QAnywhere の追加機能を示します。

新しい QAnywhere クライアント API

- **.NET API** 次の .NET API が追加されました。
 - CreateQAManager メソッド
 - CreateQAManager(Hashtable String)
 - CreateQATransactionalManager()
 - CreateQATransactionalManager(Hashtable String)
- **Java QAnywhere API の新しいインタフェース・クラス** 新しいクラスが追加されました。
「[QAMessageListener2 インタフェース](#)」 「[QAnywhere](#)」を参照してください。
- **C# QAnywhere API の新しいデリゲート**
 - 「[ExceptionListener デリゲート](#)」 「[QAnywhere](#)」
 - 「[ExceptionListener2 デリゲート](#)」 「[QAnywhere](#)」
 - 「[MessageListener2 デリゲート](#)」 「[QAnywhere](#)」

QAnywhere Agent の新機能

- **メッセージ・ストアとして機能する Ultra Light** QAnywhere で、メッセージ・ストアとして Ultra Light がサポートされるようになりました。「[qauagent ユーティリティ](#)」 「[QAnywhere](#)」を参照してください。
- **qastop -id オプション** このオプションを使用すると、メッセージ・ストア ID を qastop に指定して、指定したストア ID でデータベースに接続している QAnywhere Agent を停止できます。
- **新しい qauagent ユーティリティ** Ultra Light 用の QAnywhere Agent です。「[qauagent ユーティリティ](#)」 「[QAnywhere](#)」を参照してください。

その他の QAnywhere の強化

- **メッセージ・ストアとして機能する Ultra Light** QAnywhere で、メッセージ・ストアとして Ultra Light がサポートされるようになりました。「[qauagent ユーティリティ](#)」 「[QAnywhere](#)」を参照してください。
- **インクリメンタル・アップロード／ダウンロードの強化** インクリメンタル・アップロードとインクリメンタル・ダウンロードでは、大きなメッセージを小さなメッセージに分割できます。「[-iu オプション](#)」 「[QAnywhere](#)」と「[-idl オプション](#)」 「[QAnywhere](#)」を参照してください。

- **設定プロパティとしてのデータベース・タイプの指定** QAnywhere Manager の設定プロパティとしてクライアント・メッセージ・ストアのデータベース・タイプを指定できるようになりました。「[QAnywhere Manager の設定プロパティ](#)」『[QAnywhere](#)』を参照してください。
- **最終ステータス到達時におけるメッセージのアーカイブ** メッセージが最終ステータスになり、永久的に削除されるまで待機しているメッセージを格納するためだけに使用されるアーカイブ・メッセージ・ストアが用意されました。「[アーカイブ・メッセージ・ストア要求](#)」『[QAnywhere](#)』を参照してください。
- **サーバ側の削除ルール** サーバ側の削除ルールがアーカイブ・メッセージ・ストアに適用されるようになりました。
- **サーバ管理要求の強化** サーバ管理要求が拡張され、<actions> タグ内にオプションで追加可能な <actionsResponseId> タグが含まれるようになりました。

QAnywhere の動作の変更と廃止予定機能

次に、バージョン 11.0.0 で導入された QAnywhere の変更点を示します。

QAnywhere Agent の変更

- **qastop ユーティリティ** qastop にオプションが指定されていないと、同じコンピュータ上で実行中のすべての QAnywhere Agent が停止します。
- **Windows Mobile の予約スタック・サイズ** Windows Mobile のための、*qaagent.exe*、*dblsn.exe*、*dbmlsync.exe* の全スレッドの予約スタック・サイズが縮小されました。

QAnywhere のその他の変更

- **非ブロッキング・ダウンロード確認** 正常にダウンロードされたメッセージのステータスは、同期の直後にサーバ内で「転送中」から「転送済み」に更新されます。

SQL Remote

次の項では、SQL Remote バージョン 11.0.0 の新機能、動作の変更、廃止予定機能について説明します。

SQL Remote の新機能

- **抽出ユーティリティ (dbxtract) の -ea オプションの強化** dbxtract 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。「抽出ユーティリティ (dbxtract)」『SQL Remote』を参照してください。
- **抽出ユーティリティ (dbxtract) への -ap、-er、-et、-nl オプションの追加** dbxtract で -ap、-er、-et、-nl の各オプションを使用できるようになりました。「抽出ユーティリティ (dbxtract)」『SQL Remote』を参照してください。

SQL Remote の動作の変更と廃止予定機能

次に、バージョン 11.0.0 で導入された SQL Remote の変更点を示します。

- **挿入操作でロードされたローをロギングする LOAD TABLE のオプション機能** 新しい WITH ROW LOGGING 句によって、SQL Anywhere リモート・データベースで LOAD TABLE 文を使用してデータをロードできるようになりました。これまでは、LOAD TABLE ではローのログが取られずに dbremote または dbltm から無視されていたため、LOAD TABLE は同期環境やレプリケート環境において必ずしも有益ではありませんでした。「LOAD TABLE 文を使用したデータのインポート」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- **VIM と MAPI の各メッセージ・タイプのサポート終了** このリリースでは、VIM と MAPI の各メッセージ・システムがサポートされなくなりました。VIM または MAPI を使用するデータベースを SQL Anywhere バージョン 11.0.0 にアップグレードする場合は、メッセージ・タイプを File、FTP、または SMTP に変更してください。メッセージ・タイプが MAPI または VIM の場合、dbremote.exe は起動しません。

最も簡単な変更方法は、SMTP/POP に切り替えることです。メッセージ・タイプを変更すると、SMTP/POP をサポートするためにメール・サーバの変更が必要になる場合があります。

SQL Remote メッセージ・タイプの選択については、「SQL Remote メッセージ・システム」『SQL Remote』を参照してください。

Ultra Light

次の項では、Ultra Light バージョン 11.0.0 の新機能、動作の変更、廃止予定機能について説明します。

Ultra Light の新機能

次に、バージョン 11.0.0 で導入された Ultra Light の追加機能を示します。

主な機能

- **同期プロファイルのサポート** Ultra Light で同期プロファイルがサポートされるようになりました。同期プロファイルを使用して、Ultra Light データベースが Mobile Link と同期する方法を定義できます。[「Ultra Light CREATE SYNCHRONIZATION PROFILE 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 SELECT 文** Ultra Light の SELECT 文に明示的な FOR 句がない場合、デフォルトで FOR READ ONLY になります。この変更によって、Ultra Light では、更新が許可されていないクエリにより適したプランを選択できるようになります。[「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。
- **Ultra Light SYNCHRONIZE 文** Ultra Light 同期プロファイルまたは特定の同期オプションを同期するための新しい文です。[「Ultra Light SYNCHRONIZE 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 CREATE SYNCHRONIZATION PROFILE 文** Ultra Light 同期プロファイルを作成するための新しい文です。同期プロファイルによって、Ultra Light データベースが Mobile Link サーバと同期する方法を定義します。[「Ultra Light CREATE SYNCHRONIZATION PROFILE 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 ALTER SYNCHRONIZATION PROFILE 文** Ultra Light 同期プロファイルを変更するための新しい文です。[「Ultra Light ALTER SYNCHRONIZATION PROFILE 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 DROP SYNCHRONIZATION PROFILE 文** Ultra Light 同期プロファイルを削除するための新しい文です。[「Ultra Light DROP SYNCHRONIZATION PROFILE 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。
- **SQL Anywhere パススルー・スクリプトのサポート** Ultra Light ユーティリティで、SQL Anywhere パススルー・スクリプトがサポートされるようになりました。この変更点は次のユーティリティに適用されます。
 - ulcond11
 - ulunload
 - ulload
 - ulinfo
 - ulsync

次の項を参照してください。

- 「SQL パススルーの Ultra Light サポート」 『Ultra Light データベース管理とリファレンス』
- 「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 (ulcond11)」 『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』
- 「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 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』
- 「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 データベース検証** ulvalid ユーティリティまたは ValidateDatabase API を使用して Ultra Light データベースを検証できるようになりました。検証によって、データベース・ファイル内に特定タイプの破損がないかテストされます。コマンド・ライン・パラメータを使用して結果を絞り込むことができます。「Ultra Light データベース検証ユーティリティ (ulvalid)」 『Ultra Light データベース管理とリファレンス』と「Ultra Light データベースの検証」 『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 .NET で ValidateDatabase 関数がサポートされるようになりました。接続の有無に関わらず、データベースや特定のテーブルを検証できるようになりました。「ULDatabaseManager クラス」 『Ultra Light - .NET プログラミング』と「ULConnection クラス」 『Ultra Light - .NET プログラミング』を参照してください。

Sybase Central の **データベース検証ウィザード**を使用して、Ultra Light データベースを検証できるようになりました。[データベースの検証] オプションは、[ツール] メニューから選択できます。

- **イベントと通知のサポート** Ultra Light でイベントと通知がサポートされるようになりました。イベントの発生時に、通知メッセージが登録されたキューまたは接続に送信されます。アプリケーションによるユーザ・イベントの定義やトリガも可能です。イベントと通知の API は、サポートされている各言語で提供されます。また、API 機能にアクセスするための SQL 関数も提供されています。
- **Ultra Light における独立性レベルのサポート** 接続が複数ある場合、デフォルトで互いに独立するようになりました。他の接続によるコミットされていない変更およびダウンロードは、コミットされるまで参照できません。

独立性レベルを READ_COMMITTED または READ_UNCOMMITTED に設定できるようになりました。「Ultra Light の独立性レベル」 『Ultra Light データベース管理とリファレンス』と「Ultra Light でのトランザクション処理」 『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 .NET で ReadUncommitted 独立性レベルがサポートされるようになりました。オートコミット・モードでの接続の独立性レベルは、デフォルトで ReadCommitted になります。「Ultra Light でのトランザクション処理」 『Ultra Light データベース管理とリファレンス』と「Ultra Light の独立性レベル」 『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 ALTER DATABASE SCHEMA FROM FILE 文** ALTER DATABASE SCHEMA FROM FILE 文を使用して Ultra Light スキーマを変更できるようになりました。ALTER DATABASE SCHEMA FROM FILE 文は、現在は削除されている UpgradeSchemaFromFile メソッドまたは ApplyFile メソッドによって実装されたバージョン 9.0.2 におけるスキーマ・アップグレード機能を置き換えるものとなります。ulinit ユーティリティまたは ulunload ユーティリティのいずれかを使用して、必要な DDL 文が構文的に正しいことを確認してください。
次の項を参照してください。
 - 「Ultra Light ALTER DATABASE SCHEMA FROM FILE 文」 『Ultra Light データベース管理とリファレンス』
 - 「Ultra Light スキーマのアップグレードの配備」 『Ultra Light データベース管理とリファレンス』
 - 「Ultra Light データベース初期化ユーティリティ (ulinit)」 『Ultra Light データベース管理とリファレンス』
 - 「Ultra Light データベースのアンロード・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』
- **データベース抽出ウィザードの動作の変更** 抽出プロセスからテーブルを除外できるようになりました。データベース抽出ウィザードでは、使用可能なパブリケーションのリストから重複する名前の付いたパブリケーションが省略されます。「バージョン 9 以前の Ultra Light データベースのアップグレード」 420 ページを参照してください。
- **Ultra Light クライアント・バージョンとビルド番号の Mobile Link ログ・ファイルへの追加** 同期中に、Ultra Light クライアントがバージョンとビルド番号を Mobile Link サーバ・ログに追加するようになりました。次の項を参照してください。
 - 「Mobile Link サーバ・ログの表示」 『Mobile Link - サーバ管理』
 - 「Ultra Light の同期パラメータ」 『Ultra Light データベース管理とリファレンス』
- **Ultra Light LOAD TABLE 文** LOAD TABLE 文をデスクトップ・コンピュータ上で実行できるようになりました。Ultra Light LOAD TABLE 文を参照してください。「Ultra Light LOAD 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。
- **バックグラウンド同期のサポート** アプリケーションのどの時点からでも別のスレッドの同期を開始できるようになりました。Ultra Light では、アップロードの開始時にコミットされたローのみがアップロードされます。アップロード中にデータベースを変更し、アップロードに影響を及ぼさずにその変更をコミットできるようになりました。アップロード中にコミットされたローは、アップロードからは無視されます。「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。
- **強化された GUID 識別子のサポート** Ultra Light のこれまでのバージョンでは、16 バイトのバイナリまたは文字列で表した UUID (ユニバーサル・ユニーク識別子) または GUID (グローバル・ユニーク識別子) 識別子の入出力がランタイムにより許可されていました。これらの識別子はエンディアン変換によって GUID 構造体と互換にされていました。Ultra Light 11 では、GUID 構造体は、ランタイムから明示的に入出力できるため、エンディアン変換が不要になりました。
- **ul_stream_error 構造体** Ultra Light 11 では、stream_id フィールド、stream_context フィールド、および error_string_length フィールドが削除されました。また、error_string フィールド

が、ユーザ定義の `char *` から静的 `char` 配列に変更されました。「[ul_stream_error 構造体](#)」
『[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

- **64 ビット Windows プラットフォーム (64 ビットの XP 以降) へのネイティブ amd64/x64 ESQL および C++ アプリケーション配備のサポート** Ultra Light で、64 ビット Windows プラットフォーム (64 ビットの XP 以降) に対するネイティブ amd64/x64 ESQL および C++ 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。ただし、64 ビットのコンピュータで Ultra Light アプリケーションを開発するには、Ultra Light のユーティリティの 32 ビット・バージョンを使用する必要があります。また、64 ビットのコンピュータで Ultra Light に接続する場合は、DBISQL と Sybase Central の 32 ビット・バージョンを使用する必要があります。
- **Linux (32 ビット) にポートされた Ultra Light ユーティリティ** 「[ユーティリティ](#)」 84 ページを参照してください。

セキュリティ

- **エンドツーエンド暗号化** Ultra Light で、プロトコルレベルでエンドツーエンド暗号化がサポートされるようになりました。データは、暗号ブロック連鎖 (CBC) モードで 128 ビット AES を使用して暗号化され、キー交換は RSA または ECC によって処理されます。

ユーティリティ

- **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)** `-s` オプションを使用してスキーマをアンロードし、SQL Anywhere 互換のフォーマットでデータを保存できるようになりました。「[Ultra Light データベースのアンロード・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。
- **Ultra Light データベース初期化ユーティリティ (ulinit)** `-d` オプションを使用して SQL Anywhere データベースから新しい Ultra Light データベースにデータをコピーできるようになりました。「[Ultra Light データベース初期化ユーティリティ \(ulinit\)](#)」 『[Ultra Light データベース管理とリファレンス](#)』を参照してください。
- **ulerase** Ultra Light データベース (データベースに関連するテンポラリ・ファイルを含む) を消去するための新しいユーティリティ・プログラムです。このユーティリティには、データベースへのアクセスを確認するためのユーザ ID とパスワードが必要です。「[Ultra Light データベース消去 \(ulerase\)](#)」 『[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。
- **ulvalid** Ultra Light データベースを検証するための新しいユーティリティ・プログラムです。検証によって、データベース・ファイル内に特定タイプの破損がないかテストされます。コマンド・ライン・パラメータに従って検証を設定できます。「[Ultra Light データベース検証ユーティリティ \(ulvalid\)](#)」 『[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。

● Linux にポートされた Ultra Light ユーティリティ (32 ビット・バージョンのみで使用可能)

- **ulcreate** 新しい空の Ultra Light データベースを作成します
- **ulerase** Ultra Light データベースと関連するチェックポイント・ファイルを永久的に消去します
- **ulinfo** 既存の Ultra Light データベースに関する情報を表示します
- **ulinit** SQL Anywhere リファレンス・データベースで使用可能なスキーマから新しい Ultra Light データベースを作成します
- **ulload** ulunload によって保存された XML データから Ultra Light データベースを作成し、ロードします
- **ulsync** Mobile Link を転送エージェントとして使用して、Ultra Light データベースを統合データベースと同期します
- **ulunload** Ultra Light データベースを XML にアンロードします
- **ulvalid** Ultra Light データベースに妥当性検査を実行します

ulunloadold ユーティリティは Linux では使用できません。

SQL

- **IF 文、CASE 文、CASE 式の強化** 互換性の向上のため、IF 式を ENDIF または END IF で終了できるようになりました。また、CASE 式を END または END CASE で終了できるようになりました。[「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。

プログラミング・インタフェース

一般的な向上点

パブリケーション・マスクは、パブリケーション・リストに置き換えられました。キーワード Publications は、パブリケーション名をカンマで区切ったリストで指定します。

Ultra Light for C/C++

新しいメソッド、UltraLite_Table* OpenTableEx() は、UltraLite_Connection オブジェクトの一部になりました。このメソッドによって、SQL 以外のアプリケーションからテーブルを開いたり、ローのダイレクト・スキャンを実行する際の汎用性が向上します。[「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』](#)を参照してください。

このメソッドを使用すると、次のいずれかの方法を指定してテーブルを開くことができます。

- プライマリ・キーのローを返すには、ul_table_open_primary_key を使用します。
- 任意の順序でローを返すには、ul_table_open_no_index を使用します。

- インデックスによって指定された順序でローを返すには、`ul_table_open_with_index` を使用します。

このメソッドの完全な構文とリファレンスについては、「[OpenTableEx 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

- エラー解釈に関連する 2 つの関数のマニュアルについては、「[ULGetErrorParameter 関数](#)」『[Ultra Light - C/C++ プログラミング](#)』と「[ULGetErrorParameterCount 関数](#)」『[Ultra Light - C/C++ プログラミング](#)』を参照してください。
- パブリケーション・マスクからパブリケーション・リストへの変更を反映するため、`ULGetLastDownloadTime`、`ULResetLastDownloadTime`、および `ULCountUploadRows` の各関数の構文が変更されました。
- 関数 `ULGetPublicationMask` は使用できなくなりました。

Ultra Light .NET

- `ULDataReader` クラスに、指定された最大ロー数までロー・カウントを取得できる新しいメソッド `GetRowCount(threshold)` が追加されました。
- `ULDataReader` クラスで `IListSource` インタフェースが実装されるようになりました。

Ultra Light for M-Business Anywhere

- `Connection` クラスにイベント処理と通知のための新しいメソッド `cancelGetNotification`、`createNotificationQueue`、`declareEvent`、`destroyNotificationQueue`、`getNotification`、`getNotificationParameter`、`registerForEvent`、`sendNotification`、`triggerEvent` が追加されました。

Ultra Light J

Ultra Light J は Java による Ultra Light の実装で、Java SE 環境や BlackBerry スマートフォンを含む Java ME 環境をサポートします。「[Ultra Light J の概要](#)」『[Ultra Light J](#)』を参照してください。

Ultra Light の動作の変更と廃止予定機能

次に、バージョン 11.0.0 で導入された Ultra Light の変更点を示します。

廃止予定のプラットフォーム

- Ultra Light C++ インタフェースで Symbian OS がサポートされなくなりました。Symbian 向けの Ultra Light アプリケーションの開発者は、Ultra Light J を使用してください。「[Ultra Light J の概要](#)」『[Ultra Light J](#)』を参照してください。
- Ultra Light .NET インタフェースで .NET 1.0 コンポーネントがサポートされなくなりました。.NET 1.0 API リファレンスは、マニュアルから削除されました。

バージョン 11 では、Ultra Light for Appforge はサポートされていません。

データベース・プロパティ

このリリースでは、次のデータベース・プロパティが推奨されなくなっています。

- CollationName

接続パラメータ

このリリースでは、Ultra Light の次の接続パラメータが推奨されなくなりました。

- ORDERED_TABLE_SCAN

削除されたユーティリティ

C++ API 移行ウィザードは Sybase Central で使用できなくなりました。

削除、廃止予定、変更された関数

- ESQL インタフェースの ULGetPublicationMask 関数は削除されました (パブリケーション・マスクはパブリケーション・リストに置き換えられました)。
- パブリケーション・マスクからパブリケーション・リストへの変更を反映するため、ESQL インタフェースの ULgetLastDownloadTime、ULResetLastDownloadTime、および ULCountUploadRows の各関数の構文が変更されました。
- M-Business Anywhere API で、DatabaseSchema.getTableCountInPublications、DatabaseSchema.SYNC_ALL_DB、DatabaseSchema.SYNC_ALL_PUBS、PublicationSchema.getMask、SyncParms.getPublicationMask、SyncParms.setPublicationMask の各関数は推奨されなくなりました。
- M-Business Anywhere および .NET API では、次のメソッドはパラメータとしてパブリケーション・マスクを取っていました。このリリースでは、パラメータがパブリケーション・リスト (文字列) に変更されました。影響を受けるメソッドは、Connection クラスの countUploadRows、getLastDownloadTime、および resetLastDownloadTime です。
- ul_sync_info 構造体に変更され、**disable_concurrency**、**checkpoint_store**、**table_order** の各フィールドが削除されました。これらのオプションは、**additional_parms** という新しいフィールドに、キーワードと値の組み合わせをセミコロンで区切って指定します。

その他

- **CustDB の 2 番目のインスタンスの実行** これまでのバージョンでは、CustDB の 2 番目のインスタンスを実行すると、エラーが発生しました。このバージョンでは、CustDB の 2 番目のインスタンスを起動すると、最初のインスタンスがフォアグラウンドに移され、2 番目のインスタンスは終了します。

Sybase Central と Interactive SQL

次の項では、バージョン 11.0.0 の Sybase Central と Interactive SQL での新機能、動作の変更、廃止予定機能について説明します。

Sybase Central と Interactive SQL の新機能

次に、バージョン 11.0.0 で導入された Sybase Central と Interactive SQL の追加機能を示します。

- **新しい高速ランチャ方式** これまで、Interactive SQL および Sybase Central の高速ランチャは、ユーザのログイン時に起動していました。このバージョンでは、高速ランチャは Interactive SQL または Sybase Central の起動時にのみ起動します。起動されると、高速ランチャはアプリケーションのシャットダウン後 30 分間動作するよう設定されています。この新しい方式によって、30 分以内に実行される後続の Interactive SQL および Sybase Central の高速ランチャをより短時間で起動できるようになります。「[高速ランチャ・オプションの使用](#)」『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。
- **[接続] ウィンドウの強化と変更** Sybase Central および Interactive SQL の [接続] ウィンドウが、次のように強化されています。
 - **新しい [ODBC データ・ソースとして保存] ツール** [ODBC データ・ソースとして保存] ツールを使用すると、[接続] ウィンドウで指定した接続パラメータを使用して ODBC データ・ソースを生成できます。このツールを使用するには、[接続] ウィンドウの [ツール] ボタンをクリックして、[ODBC データ・ソースとして保存] を選択し、画面の指示に従います。「[\[接続\] ウィンドウを使用した ODBC データ・ソースの作成](#)」『[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 [接続文字列をクリップボードにコピー] ツール** [接続文字列をクリップボードにコピー] ツールを使用すると、接続文字列をクリップボードにコピーして、データベース・サーバに渡されている値を確認できます。このツールを使用するには、[接続] ウィンドウの [ツール] ボタンをクリックして、[接続文字列をクリップボードにコピー] を選択します。接続文字列をテキスト・エディタに貼り付けて表示します。
 - **接続アシスタント** [接続アシスタント] ツールが、[接続] ウィンドウの右半分に表示されるようになりました。[接続アシスタント] は、データベースへの接続を支援するように設計された、ウィザードのような機能です。
 - **[詳細] タブの強化** [詳細] タブに、プロパティの表とプロパティの簡単な説明が表示されるようになりました。
 - **新しい [ネットワーク] タブ** [ネットワーク] タブを使用して、サポートされているプロトコルの共有メモリと TCP/IP のオプションを指定できます。この新しいタブが、これまで [データベース] タブの [ネットワーク上でデータベース・サーバを検索] オプションによって提供されていた機能を置き換えます。
- **キーボード・ショートカット** Sybase Central のコード・エディタと、Interactive SQL の [SQL 文] ウィンドウ枠で、キーボード・ショートカットを使用してコードとコメントのインデントを増減できるようになりました。また、キーボード・ショートカットを使用して、二重ハイフン・コメントと二重スラッシュ・コメントの両方のインジケータを追加および削除できま

す。「[Sybase Central キーボード・ショートカット](#)」 『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。

Sybase Central の新機能

次に、バージョン 11.0.0 で導入された Sybase Central のプラグインの追加機能を示します。

SQL Anywhere プラグインの新機能

- **データベースの [概要] タブ** データベース・サーバとその機能の正常性と統計情報の概要を確認できるようになりました。「[データベースの正常性と統計情報のモニタリング](#)」 『[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・ドキュメント** データベース・ドキュメント・ウィザードを使用してデータベースのドキュメントを生成できるようになりました。「[データベースのドキュメント化](#)」 『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。
- **新しいウィザード** SQL Anywhere プラグインに、次の新しいウィザードが含まれるようになりました。
 - ログイン・ポリシー作成ウィザード
 - データベース・ドキュメント・ウィザード
 - テキスト・インデックス作成ウィザード
 - テキスト設定オブジェクト作成ウィザード
 - データベース作成ウィザード
- **新しいプロパティ・ウィンドウ** SQL Anywhere プラグインに、次の機能のプロパティ・ウィンドウが含まれるようになりました。
 - テキスト・インデックス
 - テキスト設定オブジェクト
 - 外部環境
 - マテリアライズド・ビュー
 - データベース監査
- **Sybase Central 内の監査ユーザ設定** Sybase Central を使用して、データベース監査のユーザ設定を設定し、監査情報を表示できるようになりました。「[監査の制御](#)」 『[SQL Anywhere サーバ - データベース管理](#)』と「[監査情報の取り出し](#)」 『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。
- **照合の適合化設定のサポート** [データベースのプロパティ] ウィンドウの [設定] タブを使用して、照合の適合化オプションを指定できるようになりました。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を作成するときに照合の適合化オプションを指定することもできます。

Ultra Light プラグインの新機能

- **新しいウィザード** Ultra Light プラグインに、次の新しいウィザードが含まれるようになりました。
 - データベース検証ウィザード
 - 同期プロファイル作成ウィザード
- **新しいプロパティ・ウィンドウ** Ultra Light プラグインに、次の機能のプロパティ・ウィンドウが含まれるようになりました。
 - 同期プロファイル

Mobile Link プラグインの新機能

- **同期プロファイルのサポート** Sybase Central で同期プロファイルを作成し、管理できます。
- **新しいウィザード** Mobile Link プラグインに、次のウィザードが含まれるようになりました。
 - パススルー・スクリプト作成ウィザード
 - パススルー・ダウンロード作成ウィザード
- **新しいプロパティ・ウィンドウ** Mobile Link プラグインに、次の機能のプロパティ・ウィンドウが含まれるようになりました。
 - パススルー・スクリプト
 - パススルー・ダウンロード

QAnywhere プラグインの新機能

- **Ultra Light のサポート** QAnywhere プラグインで、クライアント・メッセージ・ストアとして Ultra Light がサポートされるようになりました。また、新しいメッセージ・アーカイブ機能のサポートも追加されています。[「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。

Interactive SQL の新機能

次に、バージョン 11.0.0 で導入された Interactive SQL の追加機能を示します。

- **データベース・ロックの自動解放** Interactive SQL でテーブルを表示すると、テーブルを変更しなくても、テーブルに対するスキーマ・ロックがデータベース・サーバによって作成されます。

このリリースでは、結果セットが表示されるときに作成されるデータベース・スキーマ・ロックを Interactive SQL が解放しようとしています。

結果セットを返す文を実行した後、データベースにコミットされていない変更が接続にあるかどうか Interactive SQL によって確認されます。コミットされていない変更がなかった場合は Interactive SQL によってスキーマ・ロックが解放されます。それ以外の場合、スキーマ・ロックは解放されません。つまり、データベースにコミットされていない変更がある場合、スキーマ・ロックは解放されません。

- **読み込み専用結果セットのサポート** Interactive SQL で結果セットを読み込み専用に行うようになりました。これを行うには、[オプション] - [結果] を選択して、[編集の無効化] オプションを選択します。この設定は、それ以降にフェッチされた結果に適用されます。
「Interactive SQL の結果セットからテーブル値を編集する」 『SQL Anywhere サーバ - データベース管理』を参照してください。
- Interactive SQL を展開するときは、*OEM.ini* ファイル内の `lockedPreferences` のエントリに `disableResultSetEditing` を追加することで、ユーザがこの設定を変更できないようにすることができます。「管理ツールの設定」 『SQL Anywhere サーバ - プログラミング』を参照してください。
- **SQL 文を 1 文ずつ実行する** これまでは、SQL 文を 1 文ずつ実行する場合、SQL 文を選択してから [選択の実行] を実行する操作を繰り返す必要がありました。このリリースでは、[シングル・ステップ] を実行すると、選択した文が実行され、次に実行される文が選択されます。[選択の実行] と同様に、[シングル・ステップ] は Interactive SQL の [SQL] メニューから選択できます。「Interactive SQL からの SQL 文の実行」 『SQL Anywhere サーバ - データベース管理』を参照してください。
- **お気に入りの SQL ファイルとデータベース接続** よく使用するデータベース接続のリストと SQL コマンド・ファイルのリストを、Interactive SQL の [お気に入り] メニューで作成および管理できるようになりました。「お気に入りの使用」 『SQL Anywhere サーバ - データベース管理』を参照してください。
- **結果セットのテーブル編集を無効にする** Interactive SQL を展開するとき、Interactive SQL からの SQL Anywhere と Ultra Light の結果セットのテーブル編集を無効にできるようになりました。「管理ツールの設定」 『SQL Anywhere サーバ - プログラミング』を参照してください。
- **新しいキーボード・ショートカット** Interactive SQL に新しいキーボード・ショートカットが追加されました。「Interactive SQL キーボード・ショートカット」 『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・アプリケーションでのオプション変更を防止するためのサポート** ユーザが Interactive SQL のオプション設定を変更しないよう、*OEM.ini* ファイルで一部の設定をロックできるようになりました。「管理ツールの設定」 『SQL Anywhere サーバ - プログラミング』を参照してください。
- **dbisql の新しい -version オプション** コマンド・プロンプトで `dbisql -version` と入力すると、Interactive SQL のバージョン番号が表示されます。「Interactive SQL ユーティリティ (dbisql)」 『SQL Anywhere サーバ - データベース管理』と「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。

Sybase Central と Interactive SQL の動作の変更と廃止予定機能

次に、バージョン 11.0.0 で導入された Sybase Central と Interactive SQL の変更点を示します。

- **データベース・ツールのランチャの再配備の簡素化** Sybase Central、Interactive SQL、データベース・コンソール・ユーティリティ、Mobile Link モニタの各ランチャ実行プログラムの再

配備が簡単になりました。JAR ファイルの場所に関するレジストリのエントリやセット・ディレクトリ構造が不要になりました各実行プログラムには、(*filename.exe* ファイルと同じ名前の) 対応する *filename.INI* ファイルが、*exe* ファイルと同じディレクトリに必要です。*.INI* ファイルには、ツールをロードする方法の詳細が含まれます。「[管理ツールの配備](#)」『[SQL Anywhere サーバ - プログラミング](#)』を参照してください。

- **OEM.ini [help] セクションのサポート終了** *OEM.ini* ファイルの [help] セクションはサポートされなくなりました。詳細については、「[管理ツールの設定](#)」『[SQL Anywhere サーバ - プログラミング](#)』を参照してください。

Sybase Central の動作の変更と廃止予定機能

次に、バージョン 11.0.0 で導入された Sybase Central の変更点を示します。

- **Sybase Central の設定ファイルの名前変更** *.screpository* ファイルの名前が *.screpository600* に変更されました。

SQL Anywhere プラグインの変更機能

- **プロパティ・ウィンドウの強化** 次のプロパティ・ページが更新されました。
 - [Web サービスのプロパティ] ウィンドウ
 - [ユーザのプロパティ] ウィンドウ
 - [ビューのプロパティ] ウィンドウ
 - [マテリアライズド・ビューのプロパティ] ウィンドウ
- **特定ユーザのデバッグ** SQL Anywhere プラグ・インで [デバッグ] モードを選択すると、デバッグの対象となるユーザを指定する必要があります。「[チュートリアル：デバッグの使用開始](#)」『[SQL Anywhere サーバ - SQL の使用法](#)』を参照してください。
- **更新されたウィザード** 次のウィザードが更新されました。
 - ユーザ作成ウィザード
 - Web サービス作成ウィザード
 - Deployment ウィザード

Mobile Link プラグインの変更機能

- **更新されたウィザード** 次のウィザードが更新されました。
 - 同期モデル展開ウィザード

SQL Anywhere プラグインの廃止予定機能

- **削除されたプロパティ・タブ** 次のプロパティ・ウィンドウが更新されました。
 - [テーブルのプロパティ] ウィンドウ
 - [Ultra Light プロジェクトのプロパティ] ウィンドウ
 - [Ultra Light 文のプロパティ] ウィンドウ
 - [Ultra Light 文のプロパティ] ウィンドウ

Interactive SQL の動作の変更と廃止予定機能

次に、バージョン 11.0.0 で導入された Interactive SQL の変更点を示します。

- **プラン・ビューでグラフィカル・プランを表示可能** Interactive SQL で、プラン・ビューと呼ばれるサイズ変更可能な別ウィンドウで SQL Anywhere データベースのグラフィカルなプランを表示できるようになりました。この変更によって、[プラン・ビュー] ウィンドウを一度に複数開くことができるので、プランの表示と比較が簡単になります。プラン・ビューにアクセスするには、[ツール] - [プラン・ビューを開く] を選択します。Ultra Light データベースのテキスト・プランもプラン・ビューに表示されます。[「Interactive SQL のプラン・ビューによるプランの表示」](#) 『SQL Anywhere サーバ - データベース管理』を参照してください。

また、Interactive SQL の isql_plan オプションはサポートされていません。

- **長いプランと短いプランの表示はサポート終了** SQL Anywhere データベースのテキスト・プランを Interactive SQL で表示できなくなりました。ただし、EXPLANATION 関数と PLAN 関数を使用してテキスト・プランを取得することはできます。Ultra Light データベースのテキスト・プランは、Interactive SQL でプラン・ビューを使用して表示できます。
- **実行プランと結果セットの印刷** [Ctrl+P] を押すか、[ファイル] - [印刷] を選択して、[SQL 文] ウィンドウ枠の内容と結果セットを印刷できるようになりました。これまでは、[SQL 文] ウィンドウ枠の内容しか印刷できませんでした。プラン・ビューで印刷するには、[印刷] ボタンをクリックします。[「SQL 文、実行プラン、結果セットの印刷」](#) 『SQL Anywhere サーバ - データベース管理』を参照してください。
- **[SQL 文] ウィンドウ枠への行番号の追加** [SQL 文] ウィンドウ枠の左側に行番号が表示されるようになりました。この行番号によって、構文エラーの場所を特定できるようになります。
- **[SQL 文の実行] ツールバー・ボタンの強化** これまでは、[Interactive SQL] ツールバーで、[SQL 文の実行] ボタンをクリックするとすべての SQL 文が実行されていました。このリリースでは、ボタンのクリック時に、すべての文を実行するか選択されている文のみを実行するかを指定できます。

[SQL 文の実行] ボタンの動作を設定するには、[ツール] - [オプション] - [ツールバー] を選択します。[「Interactive SQL からの SQL 文の実行」](#) 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **バッチ文の実行の強化**

- Interactive SQL で文のバッチが実行されるときフィードバックが改善されました。[SQL 文] ウィンドウ枠から SQL 文を実行する場合、実行されている文が選択され、全体が見えるようにスクロールされます。[ファイル] - [スクリプトの実行] を選択してスクリプト・ファイルを実行する場合、スクリプトの進行状況を示すステータス・ウィンドウが表示されます。[「複数の SQL 文の実行」](#) 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **[結果] ウィンドウ枠の強化**

- [結果] ウィンドウ枠で、[Ctrl+A] を押すと、すべての結果を選択できるようになりました。現在フェッチされている結果だけでなく、結果セット全体も選択できます。[結果] ウィンドウ枠に結果セット全体が含まれていない場合、残りの結果をフェッチするよう要

求されます。残りをフェッチしないと、現在フェッチされている結果のみが選択されます。

- **[結果]** タブからセルをコピーすると、コピーされたデータが `isql_field_separator`、`isql_quote`、`isql_escape_character` の各 Interactive SQL オプションに基づいてフォーマットされるようになりました。また、結果セットから選択した値、ロー、およびカラムをクリップボードにコピーすることもできます。「[Interactive SQL の結果セットからローをコピーする](#)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **[結果]** タブのカラム・ヘッダをクリックすると、そのカラムを基準にして結果がソートされます。**[結果]** ウィンドウ枠に結果セット全体が含まれていない場合、残りの結果をフェッチするよう要求されます。残りをフェッチしないと、現在フェッチされている結果のみがソートされます。
- 結果セットの選択されたローに基づいて INSERT 文、DELETE 文、UPDATE 文を生成し、クリップボードにコピーできるようになりました。「[結果セットからの SQL 文の生成](#)」『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 **[結果]** ウィンドウ枠が強化され、次の機能が追加されました。これらの機能は右クリック・メニューから選択できます。
 - **[コピー] - [セルのコピー]** 選択されたセルの内容をコピーします。
 - **[コピー] - [カラムのコピー]** 選択されたセルのカラム値をコピーします。
 - **[生成] - [INSERT 文]** 選択されている各ローに対して INSERT 文を生成し、クリップボードにコピーします。
 - **[生成] - [DELETE 文]** 選択されている各ローに対して DELETE 文を生成し、クリップボードにコピーします。
 - **[生成] - [UPDATE 文]** 選択されている各ローに対して UPDATE 文を生成し、クリップボードにコピーします。生成された文によって、カラムの値が現在の値に設定されます。したがって、文を実行しても、実際にはカラムの値は変更されません。この機能は、実行に先立って、編集可能なテンプレートの UPDATE 文を提供する際に便利です。

「[Interactive SQL の結果セットからローをコピーする](#)」『SQL Anywhere サーバ - データベース管理』と「[結果セットからの SQL 文の生成](#)」『SQL Anywhere サーバ - データベース管理』を参照してください。

● Interactive SQL 文の強化

- **DESCRIBE 文の強化** DESCRIBE 文によって、Interactive SQL に接続されているデータベースまたはデータベース・サーバに関する情報を返せるようになりました。「[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。
- **INPUT 文と READ 文の強化** INPUT 文と READ 文で相対パスの解決を試行する際に、2つの方法が取られるようになりました。「[INPUT 文 \[Interactive SQL\]](#)」『SQL Anywhere サーバ - SQL リファレンス』と「[READ 文 \[Interactive SQL\]](#)」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- **INPUT 文と OUTPUT 文の強化**

- **ODBC ソースからのインポートとエクスポートの新規サポート** INPUT 文と OUTPUT 文を使用してデータベースからインポートしたりデータベースにエクスポートしたりするときに、ODBC データ・ソースを指定できるようになりました。これを行うには、新しい USING 句を使用します。「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。

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も ODBC データ・ソースを指定できます。「インポート・ウィザードを使用したデータのインポート」『SQL Anywhere サーバ - SQL の使用法』と「エクスポート・ウィザードを使用したデータのエクスポート」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **バイト順マーク (BOM) の新規サポート** データ中のバイト順マーク (BOM) を処理するかどうかを制御できるようになりました。これを行うには、新しい BYTE ORDER MARK 句を使用します。「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。

- **INPUT 文のサポート・フォーマットの変更** INPUT 文では、dBase、Lotus、Excel および FoxPro の各ファイル・フォーマットがサポートされなくなりました。TEXT と FIXED は引き続きサポートされています。サポートされなくなったファイル・フォーマットを引き続き使用したい場合は、ODBC ドライバを使用する必要があります。「INPUT 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 文のサポート・フォーマットの変更** OUTPUT 文では、dBase、Lotus、Excel および FoxPro の各ファイル・フォーマットがサポートされなくなりました。TEXT、FIXED、HTML、SQL、および XML は引き続きサポートされています。「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。

- **INPUT 文と OUTPUT 文で ASCII フォーマットの名前を TEXT に変更** INPUT 文と OUTPUT 文で TEXT が使用されるようになりました。ただし、下位互換のために ASCII の使用はまだサポートされています。

- **インポート・ウィザードとエクスポート・ウィザードの変更** インポート・ウィザードまたはエクスポート・ウィザードが終了するとき、ウィザードによって生成された SQL 文がコマンド履歴に保存されます。生成された SQL 文を表示するには、[SQL] - [履歴] を選択します。

● Interactive SQL オプション

- **isql_allow_read_client_file と isql_allow_write_client_file** これらの 2 つのオプションは、クライアント側のファイルの読み込み／書き込み要求に対する Interactive SQL の応答方法を記述します。「isql_allow_read_client_file オプション [Interactive SQL]」『SQL Anywhere サーバ - データベース管理』と「isql_allow_write_client_file 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。

- **-codepage オプションは廃止予定** Interactive SQL に特定のコード・ページを含むファイルを読み込ませる場合、INPUT 文、OUTPUT 文、または READ 文の ENCODING 句を使用してください。次の項を参照してください。
 - 「Interactive SQL ユーティリティ (dbisql)」 『SQL Anywhere サーバ - データベース管理』
 - 「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』
 - 「INPUT 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
 - 「OUTPUT 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
 - 「READ 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- **isql_plan オプションのサポート終了** Interactive SQL の isql_plan オプションはサポートされなくなりました。下位互換のため、このオプションを設定しようとすると通知されずに無視されます。「Interactive SQL のプラン・ビューによるプランの表示」 『SQL Anywhere サーバ - データベース管理』を参照してください。
- **SET OPTION 文の PUBLIC キーワードの削除** SET OPTION 文を使用して Interactive SQL のオプションを設定するための PUBLIC キーワードはサポートされなくなりました。「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。
- **Interactive SQL ランチャの変更** これまでは、Windows バージョンの Interactive SQL ランチャ実行プログラムの名前は、*dbisqlg.exe* で、コマンド・ライン・バージョンの名前は、*dbisql.exe* でした。このリリースでは、Windows バージョンの Interactive SQL は *dbisql.exe* に、コマンド・ライン・バージョンは *dbisql.com* になりました。

バッチ・スクリプトでは、*dbisql.exe* ではなく、*dbisql* または *dbisql.com* を呼び出す必要があります。

SQL Anywhere コンソール・ユーティリティの動作の変更

次に、バージョン 11.0.0 で導入された SQL Anywhere コンソール・ユーティリティの変更点を示します。

- **新しいコンソール・オプション** 日付と時刻を指定して、[オプション] ウィンドウからデータベース・サーバを停止できるようになりました。[ファイル] - [オプション] - [コンソール] を選択します。

Mobile Link モニタの動作の変更

次に、バージョン 11.0.0 で導入された Mobile Link モニタの変更点を示します。

Mobile Link モニタでは、バージョン 9 以前の Mobile Link サーバで作成されたモニタ・ファイルを読み込むことができません。Mobile Link モニタは、同じメジャー・バージョンの Mobile Link サーバだけと使用してください。また、ワーク・カラムが削除され、Mobile Link モニタの次のプロパティの名前が変更されました。

古いプロパティ名	新しいプロパティ名
preload_upload	sync_request
verify_upload	authenticate_user

マニュアルの強化

- **マニュアル・ディレクトリの強化** これまで、マニュアルのディレクトリは *install-dir¥docs* でしたが、*install-dir¥documentation* に変更になりました。個々の HTML ヘルプ・ファイルと PDF ファイルのファイル名も更新されました。

HTML ヘルプの新しいファイル名は次のとおりです。

- *dbadmin_ja11.chm*
- *dbprogramming_ja11.chm*
- *dbreference_ja11.chm*
- *dbusage_ja11.chm*
- *mlclient_ja11.chm*
- *mlserver_ja11.chm*
- *mlsisync_ja11.chm*
- *mlstart_ja11.chm*
- *qanywhere_ja11.chm*
- *sachanges_ja11.chm*
- *saerrors_ja11.chm*
- *saintro_ja11.chm*
- *scplugin_ja11.chm*
- *sqlanywhere_ja11.chm*
- *sqlremote_ja11.chm*
- *uladmin_ja11.chm*
- *ulc_ja11.chm*
- *uldotnet_ja11.chm*
- *ulj_ja11.chm*
- *ulmbus_ja11.chm*

- **サポートされているプラットフォームのページに Web サイトからアクセス可能** サポートされているプラットフォームに関する情報は、iAnywhere の Web サイト <http://www.ianywhere.jp/sas/os.html> にあります。
- **[Locate] ボタン** HTML ヘルプ・ブラウザで、**[Locate]** ボタンを使用して、現在のヘルプ・ページが目次のどこに含まれているかを確認できるようになりました。

製品全体の機能

次の項では、SQL Anywhere バージョン 11.0.0 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、廃止予定機能について説明します。

製品全体の新機能

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。

- **エラー・レポートの強化** Windows、Windows Mobile、Linux では、エラー・レポートが生成されるとウィンドウが表示されます。iAnywhere に送信するか選択する前に、このウィンドウでエラー・レポートの内容を確認できます。「[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。

-ce オプションを使用してサポート・ユーティリティ (dbsupport) を設定できるようになりました。このオプションを使用すると、dbsupport でアプリケーションを監視しているときにアプリケーションがクラッシュすると、電子メールが送信されます。「[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。
- **追加の Windows Mobile プラットフォームのサポート** SQL Anywhere で、スマートフォン用 Windows Mobile 5 と Windows Mobile 6 Standard エディションがサポートされるようになりました。Windows Mobile での SQL Anywhere サーバの実行については、「[SQL Anywhere for Windows Mobile](#)」 『[SQL Anywhere サーバ - データベース管理](#)』と「[インストール時の考慮事項：Windows Mobile 5.0 for Smartphone に関する制限](#)」 『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。
- **SQL Anywhere サンプル・データベース (demo.db) の新しいテーブル** SQL Anywhere サンプル・データベースに新しいテーブル MarketingInformation が追加されました。このテーブルの各ローには、Products テーブル内の製品を説明する HTML ページが格納されています。このテーブルが追加されたことで、機能をテストしたり試したりするときにより複雑な文字データに対してクエリを実行できます。「[SQL Anywhere サンプル・データベース](#)」 『[SQL Anywhere 11 - 紹介](#)』を参照してください。

製品全体の動作の変更

次に、バージョン 11.0.0 で導入された製品全体の変更点を示します。

- **Windows CE から Windows Mobile への変更** Windows CE の名前を使用した方が正確である場合を除き、マニュアルとソフトウェアで Windows CE という名前が Windows Mobile に変更されました。
- **readcert、gencert、reqtool の削除** ユーティリティの readcert、gencert、および reqtool は削除されました。これらは、これまで廃止予定になっていました。これらに代わり、createcert と viewcert を使用できます。「[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。

- **createcert と viewcert の各ユーティリティの Mac OS X でのサポート** 証明書ユーティリティの createcert と viewcert が、Mac OS X でサポートされるようになりました。「[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。
- **certificate と certificate_password の各プロトコル・オプションの名前変更** TLS および HTTPS の certificate と certificate_password の各プロトコル・オプションは、それぞれ identity と identity_password に名前が変更されました。次の項を参照してください。
 - SQL Anywhere データベース・サーバ:「[-ec サーバ・オプション](#)」『[SQL Anywhere サーバ - データベース管理](#)』
 - SQL Anywhere Web サーバ:「[-xs サーバ・オプション](#)」『[SQL Anywhere サーバ - データベース管理](#)』
 - SQL Anywhere プロトコル・オプション:「[Identity プロトコル・オプション](#)」『[SQL Anywhere サーバ - データベース管理](#)』と「[Identity_Password プロトコル・オプション](#)」『[SQL Anywhere サーバ - データベース管理](#)』
 - Mobile Link サーバ:「[-x オプション](#)」『[Mobile Link - サーバ管理](#)』
- **サンプル ID ファイルの変更** サンプル証明書を含む ID ファイルと、対応する TLS のプライベート・キーを含む ID ファイルの名前が、このリリースでそれぞれ変更されています。ファイル *rsaserver.crt* の名前は *rsaserver.id* に、ファイル *sample.crt* の名前は *eccserver.id* に変更されました。これらの ID ファイルのパスワードはともに、**tJ1#m6+W** から **test** に変更されました。
- **インストール・ディレクトリの変更** このリリースでは、32 ビットのソフトウェアのインストール先が *win32* ディレクトリから *bin32* ディレクトリに、64 ビットのソフトウェアのインストール先が *X64* ディレクトリから *bin64* ディレクトリに変更されました。たとえば、これまでのバージョンで *C:\Program Files\SQL Anywhere 11\win32* にインストールされたソフトウェアは、このリリースでは *C:\Program Files\SQL Anywhere 11\bin32* にインストールされます。
- **サンプル・データベースの ODBC データ・ソースの変更** これまでのリリースでは、ソフトウェアとともにインストールされたサンプル・データベースの ODBC データ・ソースはユーザ・データ・ソースでした。このリリースでは、SQL Anywhere 11 Demo、SQL Anywhere 11 CustDB、QAnywhere 11 Demo の各データ・ソースがシステム・データ・ソースになりました。
- **.NET 1.0 のサポート終了** SQL Anywhere 11 では Visual Studio 2003 はサポートされません。Visual Studio 2005 (.NET 2.0) と Visual Studio 2008 (.NET 3.x) はサポートされています。

バージョン 10.0.1 の新機能

目次

SQL Anywhere	103
Mobile Link	115
QAnywhere	118
SQL Remote	120
Ultra Light	121
製品全体の機能	123

SQL Anywhere

- 「新機能」 103 ページ
- 「動作の変更と廃止予定機能」 109 ページ

Mobile Link

- 「新機能」 115 ページ
- 「動作の変更と廃止予定機能」 116 ページ

QAnywhere

- 「新機能」 118 ページ
- 「動作の変更と廃止予定機能」 119 ページ

Ultra Light

- 「新機能」 121 ページ
- 「動作の変更と廃止予定機能」 122 ページ

SQL Remote

- 「動作の変更と廃止予定機能」 120 ページ

製品全体の機能

- 「新機能」 123 ページ
- 「動作の変更」 124 ページ
- 「Windows Vista サポートの問題」 124 ページ

廃止予定機能は変更される可能性があります

廃止予定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 オプションは廃止される予定です。[「動作の変更と廃止予定機能」](#) 109 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・サーバ・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 以降のデータベースのアップグレード](#)」 405 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、その文がマテリアライズド・ビュー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 句](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』
- 「[EXCEPT 句](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』
- 「[INTERSECT 句](#)」『[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 Unicode コードポイントをさらに多く復号化でき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 のデータベースに再ロードします。「[バージョン 10 以降のデータベースの再構築](#)」 396 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・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、AccentSensitive、PunctuationSensitivity、Properties、Specification を利用できません。「[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 \[Web サービス\]](#)」 『[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 \[Web サービス\]](#)」 『[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、Mac OS X 上で SQL Anywhere データベース・サーバを 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ーの起動** 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ー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ーの起動](#)」『[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] ウィンドウに追加された [トリガ] タブ** 非マテリアライズド・ビューのプロパティ・ウィンドウに、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、テーブルと非マテリアライズド・ビュー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マテリアライズド・ビュー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、マテリアライズド・ビューの定義で使用できました。しかし、これが原因で、マテリアライズド・ビューをコストベースの最適化で使う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マテリアライズド・ビュー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。「[マテリアライズド・ビュー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 以降のバージョンまたは 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 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強化されています。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可能でした。

- **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・レポートを送信するウィンドウの強化** エラー・レポートを iAnywhere に送信するかどうかを確認する dbsupport ウィンドウに、[エラー・レポートの表示] ボタンが追加され、エラー・レポート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。

動作の変更と廃止予定機能

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・サーバの起動後に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。「[データベース・サーバ・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。

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、TRANSLLOG、TRANSLLOGMIRROR) 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 以降のデータベースのアップグレード」 405 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ときに DATABASE SIZE の値を指定する場合、オプションだった測定単位の指定は必須になりました。「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** 以前は、SACommand.UpdatedRowSource のデフォルト値は UpdatedRowSource.Both ですが、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認証文が適用されるように、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ています。「[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 ドライバに渡される 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、**jdbc:ianywhere:** で始まる URL ヘッダも使用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す。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る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** 以前のリリースの SQL Anywhere OEM 版では、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 と指定して定義された 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 と指定して定義された 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。

ON COMMIT DELETE ROWS を指定して定義された GLOBAL TEMPORARY TABLE にはロードできません。

- **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、照合の適合理化構文や、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。

廃止予定機能とサポート終了機能

- **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る予定です。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、可能な限り 1 CPU 上で要求が実行されます。また、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 -gx サーバ・オプションも廃止される予定です。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 句は廃止される予定です。「[CREATE DATABASE 文](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』を参照してください。

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](#)」 [123 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** 新しいウィンドウが追加され、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ディスクへのページングが減り、パフォーマンスが向上します。

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。[新しいリモート・テーブル] ウィンドウを使用して、統合データベースのテーブルと同じ名前とカラムを持つ新しいリモート・

テーブルを作成できます。このウィンドウを使用して、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」 123 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』

動作の変更と廃止予定機能

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 は廃止される予定です。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際の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` デリゲートが .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 デリゲート](#)」 [『QAnywhere』](#)
- 「[ExceptionListener2 デリゲート](#)」 [『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` をリスナに渡すために、`QAManagerBase` のグローバル・インスタンスを参照したり、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 デリゲート](#)」 [『QAnywhere』](#)
- Java API : 「[QAMessageListener2 インタフェース](#)」 [『QAnywhere』](#)

動作の変更と廃止予定機能

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。

動作の変更と廃止予定機能

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る予定です。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。「[SQL Remote メッセージ・システム](#)」『[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・オブ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 でのデータ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 for C/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。

動作の変更と廃止予定機能

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

- **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。
- **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-dir\ultralite\ce\arm\ips\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\sbgs2.dll` ファイル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る予定です。

`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 ～ 2048 ビットの制限がありましたが、512 ～ 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。必要に応じて、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 認定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 認定の暗号化テクノロジー](#)」 『[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 は廃止される予定です。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には、次のような問題があります。

- **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。SQL Anywhere 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、dbelevat10.exe、dblic.exe、dbsvc.exe、installULNet.exe、mlasinst.exe、SetupVSPackage.exe、ulcond10.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。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 dbelevat10.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 のサポート** 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。

バージョン 10.0.0 の新機能

目次

SQL Anywhere	129
Mobile Link	201
QAnywhere	225
SQL Remote	231
Ultra Light	233
Sybase Central と Interactive SQL	248
マニュアルの強化	254
製品全体の機能	255

SQL Anywhere

- 「新機能」 129 ページ
- 「動作の変更」 165 ページ
- 「廃止予定機能とサポート終了機能」 192 ページ

Mobile Link

- 「新機能」 201 ページ
- 「動作の変更と廃止予定機能」 214 ページ

QAnywhere

- 「新機能」 225 ページ
- 「動作の変更と廃止予定機能」 228 ページ

SQL Remote

- 「新機能」 231 ページ
- 「動作の変更と廃止予定機能」 231 ページ

Ultra Light

- 「新機能」 233 ページ
- 「動作の変更と廃止予定機能」 244 ページ

Sybase Central と Interactive SQL

- [「新機能」 248 ページ](#)
- [「動作の変更と廃止予定機能」 250 ページ](#)

マニュアルの強化

- [「マニュアルの強化」 254 ページ](#)

廃止予定機能は変更される可能性があります

廃止予定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・サーバの追加機能を示します。

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 2～3 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 [データベース] [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。
- **マテリアライズド・ビューのサポート** 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 ではマテリアライズド・ビューがサポート

されました。「マテリアライズド・ビューの操作」『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・サーバの機能が強化され、クエリの一部に応答するために使用できるマテリアライズド・ビュー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。「マテリアライズド・ビュー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。

マテリアライズド・ビュー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、Unicode 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』

Unicode 照合アルゴリズム (UCA) を使用するとき、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。

Unicode のサポートのため、International Components for Unicode (ICU) が使用され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・プロパティ」『SQL Anywhere サーバ - データベース管理』と「接続プロパティ」『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・テクノロジーを使用して、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・ログの概要」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **ロックの強化** ロックが次のように強化されています。

○ **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、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ーが廃止されました。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にも制御できます。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 の格納](#)」『[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 関数で使用する圧縮方式と同じであり、Windows の .zip ファイルで使用するアルゴリズムでもあります。「[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「[接続パラメータの矛盾の解決](#)」『[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 のアドレスを受信でき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 (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、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させるために SQL Anywhere に加えられた強化について説明します。

- **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途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 11 - 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え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 11 - 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](#)」『[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](#)」『[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](#)」『[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指定します。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 オプションにマテリアライズド・ビュー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 ({|W|E} yyyy-mm-dd hh:mm:ss message) を使用するようにしました。

- **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保存することができる -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、他のサブネット上で実行されている SQL Anywhere データベース・サーバや、ファイアウォールの外側にあつて UDP ブロードキャストが通常は届かない SQL Anywhere データベース・サーバを、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。「Broadcast Repeater ユーティリティ (dbns11)」『SQL Anywhere サーバ - データベース管理』を参照してください。
- **新しいレポート送信ユーティリティ (dbsupport)** 新しいサポート・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) の問い合わせ機能、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。「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 が次のように強化されています。
 - 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させるために、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** 再ロード時にマテリアライズド・ビューを初期化します。
 - **-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** 各テーブルに対する LOAD TABLE 文と INPUT 文を含み、データは含まない 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。
- **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マテリアライズド・ビュー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にのフォーマットを制御します。
- **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データベース・サーバ・メッセージ・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** このリリースには、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
- tsq_l_outer_joins
- verify_password_function

接続プロパティの詳細については、「[接続プロパティ](#)」『[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** このリリースには、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、「[データベース・サーバ・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・プロパティ** このリリースには、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・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** このリリースには、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

次に、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** 指定した時間中、データベース・クリーナを実行します。「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

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、指定したマテリアライズド・ビューの情報 (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マテリアライズド・ビュー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 個のビット配列を引数に取り、引数をビット処理 AND 演算した結果を返します。このとき、比較される各ビットで、両方のビットが 1 の場合は 1 を、それ以外の場合は 0 を返します。「[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 個のビット配列を引数に取り、引数をビット処理で OR 演算した結果を返します。このとき、比較される各ビットで、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 を返します。「[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 個のビット配列を引数に取り、引数をビット処理排他 OR 演算した結果を返します。このとき、比較される各ビットで、(両方のビットではなく) 片方のビットだけが 1 の場合は 1 を、それ以外の場合は 0 を返します。「[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最適化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、データベース・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** データベース・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** データベース・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 つのパラメータを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、LastIdle 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 認定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、このマニュアルで前述した機能に関する項に記載されている文の変更に追加される機能です。

- **マテリアライズド・ビューをサポートする SQL 文** マテリアライズド・ビュー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 MATERIALIZED VIEW 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、マテリアライズド・ビュー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・キーの間に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・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 句によるマテリアライズド・ビュー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。指定した文字列で始まる入力行は、ロード操作時に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 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させるために、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』、「ClusteredHashGroupBy アルゴリズム (GrByHClust)」『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 [Web サービス]」『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す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。「[iAnywhere.Data.SQLAnywhere ネームスペース \(.NET 2.0\)](#)」 『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 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。
- **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。iAnywhere JDBC ドライバで JDBC 2.0 がサポートされなくなりました。既存のアプリケーションを変更しないで引き続き実行できるように、iAnywhere.ml.jdbcodbc.IDriver クラスと iAnywhere.ml.jdbcodbc.jdbc3.IDriver クラスのサポートは継続されますが、どちらのドライバも現在は同じであり、JDBC 3.0 だけを実装します。バージョン 1.4 より前の JRE は iAnywhere JDBC ドライバとともに使用できなくなりました。「[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。
- **iAnywhere JDBC ドライバでの SQL Server Native Client ODBC ドライバのサポート** iAnywhere 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。
- **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。
- **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-nnnnnnnnnnnnn'} です。

- **ODBC メッセージ・コールバックが接続ごとになった** Adaptive Server Anywhere バージョン 9.0.0 以来、ODBC によるメッセージ・コールバックがサポートされていますが、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 DBI モジュール](#)」『[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が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

- **Deployment ウィザード** Deployment ウィザードが追加され、SQL Anywhere for Windows の配備ファイルを作成できるようになりました。Deployment 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、Deployment ウィザードを使用して、SQL Anywhere の配備ファイルを作成してください。[「Deployment 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。
- **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] ウィンドウに、Windows CE プロキシ・ポートの設定ツールが追加されています。これにより、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 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 Mobile 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** スクリプト実行ユーティリティ (dbrunsql) を使用すると、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。[「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。

UNIX/Linux の強化

- **UNIX プラットフォームで利用できる新しい 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 での SQL Anywhere 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 プロシージャからクライアント側 cookie を設定および取得する方法、HTTP Web サービス・プロシージャからバイナリ・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、タスクやコルーチン間のコンテキスト・スイッチ処理をサーバがより制御できるようになり、データベース操作との親和性が向上します。
- **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、HP-UX、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 接続は、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ます。
- **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 \[Web サービス\]](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』、
「[CREATE FUNCTION 文 \[Web サービス\]](#)」『[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、SOAP 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 \[Web サービス\]](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』、
「[CREATE FUNCTION 文 \[Web サービス\]](#)」『[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` に格納されるようになりました。

3 つの新しいシステム・テーブル `ISYSPHYSIDX`、`ISYSIDXCOL`、`ISYSFKEY` によって、`ISYSIDX` にリストされてい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、クラストード・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。

その他に、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返され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、廃止予定の 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。
- **データベース・サーバ・メッセージ・ウィンドウの強化** データベース・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、データベース・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と、[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、データベース・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 (データベース・サーバ・メッセージ・ログ・ファイル、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データベース・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、データベース・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** データベース・サーバが起動すると、デフォルトでデータベース・サーバ・メッセージ・ウィンドウは最小化されます。データベース・サーバの

起動後に、データベース・サーバ・メッセージ・ウィンドウが最小化されないにするには、`-qn` オプションを指定します。「[-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。

- **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 とミラーリング中のサーバとの接続が切断されて、別のサーバとの接続が新しく確立された場合、SQL Anywhere SNMP Extension Agent によってトラップが送信され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。この違いは要求のログのサイズが大きいとより顕著になります。

- **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・アクセスで使用する ODBC の変更](#)」258 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。「[SQLANYSAMP11 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、サポート対象または対象外のサード・パーティ・コンポーネントのバージョン番号に関する変更点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、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」 392 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、<http://www.sybase.com/products/informationmanagement/softwaredeveloperkit/jconnect> から別途ダウンロードできます。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 文を使用することはできません。以前のバージョンのデータベースをバージョン 10 にアップグレードするには、アンロードと再ロードを実行し、データベースを再構築する必要があります。「SQL Anywhere のアップグレード」 392 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。

- パスワードを最初に入力したのが大文字と小文字を区別しないデータベースだった場合、そのパスワードの大文字と小文字は区別されません。
- パスワードを最初に入力したのが大文字と小文字を区別するデータベースだった場合、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・サーバで、パスワードのハッシュ処理に SHA256 が使用され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には古い(独自の)ハッシュ処理アルゴリズムが引き続きサポートされていますが、新しいパスワードにはすべて SHA256 が使用されます。

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後で、どのユーザでもパスワードを指定することなく同じデータベースに対して 2 つ目の接続を確立できていました。今回のリリースでは、接続ごとにパスワードの指定が必要になりました。

- **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。

等号 (=) 比較と不等号 (<>) 比較の場合は、セマンティックに変更はありません。どちらの方法(ブランクの埋め込みと後続ブランクの無視)でも同じ結果になります。ただし、不等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)。以前のバージョンの SQL Anywhere 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 を返していましたが、これらは値 Yes と No を返すようになりました。「[データベース・サーバ・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。このファイルの名前は、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 Mobile 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、旧バージョン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ていると、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方法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](#)」『[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。「[データベース・サーバ・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、SAOLEDB.10 という名前で具体的に参照できます。
- **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 ですが、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。「[接続パラメータの構文ルール](#)」『[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。「[接続パラメータの構文ルール](#)」『[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。今回のバージョンでは、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、不適切な精度 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。
- **サポート対象外のデータベース・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

- **サポート対象外のプロパティ** 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。
- **サポート対象外の互換ビュー・カラム** システム互換ビューで、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・プロパティは廃止される予定です。
 - 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%\$ajvm.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** このリリースでは、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* という形式でした (*X* は PHP のメジャー・バージョン番号、*Y* は SQL Anywhere のメジャー・バージョン番号)。今回のバージョンでは、PHP モジュール・ファイルの名前は *php-a.b.c_sqlanywhereY.dll* という形式になります (*a.b.c* はファイルのビルド元 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)。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** 下位互換のため、PrefetchBuffer 接続パラメータで 16384 未満の値がキロバイトとして解釈され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 DOMAIN 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・ユーティリティは次のように変更されています。
 - **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、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 を指定したかどうかに関係なく、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` で実行中で、この IP アドレスが `1.2.3.4` の場合、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` のいずれでも使用できます。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` によって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` だけが機能していました。[「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)` で接続しようとしても機能しません。ただし、ホスト名 `localhost` または `127.0.0.1` を使用すると機能します。

- **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』](#) を参照してください。

`-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。

- 90% of (total_physical_memory - 4 MB)、2 MB 以上
- (available address space - 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 ビットの UNIX プラットフォームでは、物理メモリ量の合計の 90% または 1,834,880 KB のいずれか小さい方です。
- 64 ビットの UNIX プラットフォームでは、物理メモリ量の合計の 90% または 8,589,672,320 KB のいずれか小さい方です。

[「キャッシュ・メモリ使用の制限」](#) [『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 の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 のデータベースから抽出する必要があります。
- **Solaris でサポート対象外の -ui と -ux サーバ・オプション** 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 列挙体の変更** a_validate_type 列挙体の 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 プラットフォームの場合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の場合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の 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。

ブランクが埋め込みが有効になっているデータベースでは、ansi_blanks オプションの設定によって、フェッチする式が CHAR または NCHAR であり (VARCHAR または NVARCHAR ではない)、ホスト変数 char または nchar (VARCHAR または NVARCHAR ではない) に格納される場合に、トランケーション警告をクライアントに送信するかどうかが制御されます。「[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、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。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。

- **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 \[データベース\]」](#)
[『SQL Anywhere サーバ - データベース管理』](#)を参照してください。

- **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 は廃止される予定です。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。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' は 3 文字の月名を示しています。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-filename.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、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、廃止予定であるか、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	-
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』

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
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』
SYSCOLLATIONMAPP INGS / -	-	「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』
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』

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
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 互換ビュー (旧式)」 『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 / -	ISYSJARCOMPONEN T	「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』
- / -	ISYSMVOPTIONNAM E	「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』
SYSOPTJOINSTRATEG Y / SYSOPTJOINSTRATEG IES	-	システムでのみ使用
SYSOPTORDER / SYSOPTORDERS	-	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
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
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』

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
SYSREMOTEOPTION / SYSREMOTEOPTIONS , SYSREMOTEOPTION2	ISYSREMOTEOPTIO N	「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』
SYSREMOTEOPTIONT YPE / -	ISYSREMOTEOPTIO NTYPE	「SYSREMOTEOPTIONTYPE システム・ ビュー」 『SQL Anywhere サーバ - SQL リ ファレンス』
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 / -	ISYSSQLSERVERTYP E	「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』

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
SYSSYNC / SYSSYNCS, SYSSYNC2	ISYSSYNC	「SYSSYNC システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』、 「SYSSYNCS 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』
-	ISYSSYNCSSCRIPT	「SYSSYNCSSCRIPT システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』 と 「SYSSYNCSSCRIPTS 統合ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
- / SYSSYNCSUBSCRIPTI ONS	-	「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』
- / 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』

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
-	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』
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』

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
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 システム・ビューの各ローは、システム内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・ビューで記述されている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。
SYSLOGINMAP	SYSLOGINMAP システム・ビューには、統合化ログインまたは 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、マテリアライズド・ビューのオプション値 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・ビューで定義されている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。

システム・ビュー名	詳細な情報
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・オブジェクトの例として、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ている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 のスクリプト化されたアップロードに関するストアド・プロシージャを識別します。「 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、データベースの各テーブルとビューの各カラムのローが含まれます。「 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。
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。

システム・ビュー名	詳細な情報
SYSUSERAUTHORITY	SYSUSERAUTHORITY システム・ビューの各ローは、1 つのユーザー 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。

廃止予定のテーブルやビューの一覧

廃止予定のカatalog・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。

廃止予定のテーブルやビュー	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。

廃止予定のテーブルやビュー	変換情報
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。
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。
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 以降のカatalog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。

削除されたテーブルやビュー	変換情報
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。
- **SYSREMOTOPTION ビュー** SYSREMOTOPTION から選択できなくなりました。代わりに SYSREMOTOPTIONS または SYSREMOTOPTION2 を使用してください。「[SYSREMOTOPTIONS 統合ビュー](#)」『[SQL Anywhere サーバ - SQL リファレンス](#)』または「[SYSREMOTOPTION2 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 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。
- **SYSPROCARM 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、SYSPROCARM の width カラムが、SMALLINT から UNSIGNED INT に変更されました。「[SYSPROCARM 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。
- **派生テーブルのキー・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派生テーブルでキー・ジョインを行えなくなりました。「[ビューと派生テーブルのキー・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。結果として、AUTOINCREMENT のようなデフォルト値が、スキップされたローに対しても影響を及ぼします。AUTOINCREMENT の場合は、AUTOINCREMENT のシーケンスで値がスキップされます。以前のバージョンでは、スキップされた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 文の構文が変更され、ALTER INDEX 文の構文と一致す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、廃止される予定です。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。

廃止予定機能とサポート終了機能

- **検証ユーティリティ (dbvalid) の -f、-fi、-fd、-fn オプションは廃止予定** 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エクスプレス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る予定です。テーブルでエクスプレス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 文のオプションは廃止予定** 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る予定です。「[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 外部ジョインは廃止予定** 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・テクノロジー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。
- **サポート対象外のプロパティ** 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。「[データベース・サーバ・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 句は廃止予定** 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 は廃止される予定です。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 照合は廃止予定** UTF8 照合は廃止される予定です。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 オプションは廃止予定** 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る予定です。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はサポート対象外** 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)、DBCcollate 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 11 用に再構築」](#) 399 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 オプションはサポート対象外** 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 オプションはサポート対象外** 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 オプションはサポート対象外** 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 オプションはサポート対象外** 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・オプションはサポート対象外** 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・最適化は、テーブル内のデータ分散を基に、内部テンポラリ・テーブルのハッシュ・サイズを割り当てます。
- **max_hash_size database オプションはサポート対象外** 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・ファイルはサポート対象外** 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。
 - **読み込み専用メディアでのデータベースの展開** 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のデータベースを展開することはできます。[「読み込み専用メディアでの](#)

データベースの展開」『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。
 - データベース圧縮ウィザード
 - 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、wrtnamembuffer メンバは _unused2 として、compressed メンバは _unused3 として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** この構造体は、DBTools ライブラリを使用してデータベースを圧縮および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・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

プロシージャ名	プロシージャ名
sp_addauditrecord	sp_helpjoins
sp_addlanguage	sp_helpkey
sp_addremotelogin	sp_helplanguage
sp_addsegment	sp_helplog
sp_addserver	sp_helpremotelogin
sp_addthreshold	sp_helpprotect
sp_adddumpdevice	sp_helpsegment
sp_auditdatabase	sp_helpserver
sp_auditlogin	sp_helpsort
sp_auditobject	sp_helpthreshold
sp_auditooption	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

プロシージャ名	プロシージャ名
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
sp_dropserver	sp_unbindmsg
sp_droptreshold	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。

- **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。「[SQLANY11 環境変数](#)」『[SQL Anywhere サーバ-データベース管理](#)』と「[SQLANY11 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 プロパティは廃止予定** 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せされると常に値 On を返すようになりました。
- **サポート対象外のデータベース・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
- **サポート対象外のシステム・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 ファイルは廃止予定** 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る予定です。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。

- **廃止予定の Windows CE プラットフォーム** 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 11 - 紹介』を参照してください。

Mobile Link

次の項では、Mobile Link バージョン 10.0.0 の新機能、動作の変更、廃止予定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

synchronization model file 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。

「モデルの作成」 [『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後は、モデル・モードを使用して、同期プロジェクトをカスタマイズしてから展開できます。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することはできませんが、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。使い勝手を改善するため、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、すべての Mobile Link 統合データベースを変更できます。

「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。

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、ほぼ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、「mobile web services」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。

展開の簡略化

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展開するための展開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。

- 「Deployment 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 統合データベースに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 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。
 - 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 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。
 - 同期ステータス情報のリセット。
「[ml_reset_sync_state 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。
- **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.commmmit_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](#)
- 「[ml_qa_global_props](#)」 『[Mobile Link - サーバ管理](#)』
- [ml_qa_global_props_client](#)
- 「[ml_qa_repository](#)」 『[Mobile Link - サーバ管理](#)』
- [ml_qa_repository_client](#)
- 「[ml_qa_repository_props](#)」 『[Mobile Link - サーバ管理](#)』
- [ml_qa_repository_props_client](#)
- 「[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、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 system procedure」](#) 『[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** Mobile Link クライアントのネットワーク・プロトコル・オプションを設定する [mlsrv10 -x](#) オプションが変更されました。
[「-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。[mlsrv10](#) のインスタンス 1 つで、バージョン 10 のクライアントに加えて、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 が指定されると、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 がメッセージのタイムスタンプ設定に ISO 8601 日時フォーマットを使用** 情報、警告、エラーの各メッセージのタイムスタンプに、明確に定義された ISO 8601 日時フォーマット ({[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** スループット、柔軟性、保守性を向上させるため、Mobile Link サーバのアーキテクチャが再構成されました。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場合はデフォルトで、アップロードの場合は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 ～ 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の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がすべて表示されます。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] ウィンドウ** 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] ウィンドウ** 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 認定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](#)」 [214 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。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 11 - 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 11 - 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

- **Mobile Link サーバ・メッセージ・ウィンドウ** Linux のインストール環境には、dbmlsync と mlsrcv10 のログ情報を表示するメッセージ・ウィンドウが追加されました。
「[-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](#)」 221 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](#)」 『[Ultra Light データベース管理とリファレンス](#)』
 - 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](#)」 『[Ultra Light データベース管理とリファレンス](#)』
- **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。他のプラットフォームには以前のリリースでもこの機能が提供されていました。
- 「[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](#)」 [239 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ード・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、ローは必要に応じて動的に作成されます。

「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。
- ISYSSYNCSRIPT 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。
『SYSSYNCSRIPT 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、この動作を無効にできます。
[「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 がメッセージのタイムスタンプ設定に ISO 8601 日時フォーマットを使用** 情報、警告、エラーの各メッセージのタイムスタンプに、明確に定義された ISO 8601 日時フォーマット ({[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途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 認定のアルゴリズムを使用させることができました。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** より多く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 11 - 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。現在、TCP/IP、TLS (TLS セキュリティを使用した TCP/IP 上の同期)、HTTP、HTTPS の各プロトコルがサポートされています。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・オプションの一覧](#)」『[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・システムで信頼されている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をよりすばやく設定できるよう強化されました。
- **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。展開ウィザードは、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。

「[Notifier イベント](#)」『[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 イベント](#)」 『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 イベント](#)」 『Mobile Link - サーバ起動同期』を参照してください。

Listener 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。
- 「[Windows デバイス用の 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。
- **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。

「[Windows デバイス用の Listener ユーティリティ](#)」 『[Mobile Link - サーバ起動同期](#)』と
「[Windows 用の Listener アクション・コマンド](#)」 『[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・ハンドラでの使用に便利な新しい action 変数 \$ml_user と \$ml_password が追加されました。

「[Windows 用の Listener アクション・コマンド](#)」 『[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。

「[Windows 用の Listener アクション・コマンド](#)」 『[Mobile Link - サーバ起動同期](#)』を参照して
ください。

- **Listener がメッセージのタイムスタンプ設定に ISO 8601 日時フォーマットを使用** 情報、警告、エラーの各メッセージのタイムスタンプに、明確に定義された ISO 8601 日時フォーマット ({|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。

「[Windows デバイス用の Listener ユーティリティ](#)」 『[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。

動作の変更と廃止予定機能

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 で廃止されていましたが、このバージョンで削除されました。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。今後は、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、同期がアボートされます。
- **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・スクリプトのエラーを検出すると、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す。
- **制限が厳しくなった 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 サーバは 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、失敗した認証に係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、同期でエラーが発生した場合でも、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 以降では、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。独立性レベルはダウンロード・トランザクションの開始時に変更できますが、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。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、現在もこの方法が有効です。

「[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。

「[モデルの作成](#)」 [『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。

- **不要になったバックログ・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の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。
- 「[モデルの作成](#)」 『[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。今後は、情報、警告、エラーの各メッセージのタイムスタンプに年は常に含まれます。

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 スクリプトを使用して統合データベースに正常に挿入されたアップロード削除の総数を返します。 以前は、競合が検出されたアップロード済み削除の数を返していました。

統計プロパティ	説明
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、<code>upload_new_row_insert</code> スクリプトを使用して統合データベースに正常に挿入されたアップロード挿入の総数を返します。 以前は、競合が検出されたアップロード済み挿入の数を返していました。
conflicted_updates	通常のアップロード・モードでは、競合を引き起こ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、<code>upload_new_row_insert</code> または <code>upload_old_row_insert</code> スクリプトを使用して正常に適用されたアップロード更新ローの総数を返します。 以前は、競合が検出されたアップロード済み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、<code>handle_error</code> または <code>handle_odbc_error</code> が定義されており、1000 が返された場合、または指定のテーブルに対して <code>upload_delete</code> スクリプトが定義されていない場合において、<code>upload_delete</code> スクリプトを呼び出したときにエラーを発生させたアップロード削除ローの総数を返します。 強制的な競合モードでは、<code>handle_error</code> または <code>handle_odbc_error</code> が定義されており、1000 が返された場合、または指定のテーブルに対して <code>upload_old_row_insert</code> スクリプトが定義されていない場合において、<code>upload_old_row_insert</code> スクリプトを呼び出したときにエラーを発生させたアップロード削除ローの総数を返します。 以前は、無視されたアップロード済み削除の数を返していました。

統計プロパティ	説明
ignored_inserts	通常のアップロード・モードでは、<code>handle_error</code> または <code>handle_odbc_error</code> が定義されており、1000 が返された場合、または指定のテーブルに対して <code>upload_insert</code> スクリプトが定義されていない場合において、<code>upload_insert</code> スクリプトを呼び出したときにエラーを発生させたアップロード挿入ローの総数を返します。 強制的な競合モードでは、<code>handle_error</code> または <code>handle_odbc_error</code> が定義されており、1000 が返された場合、または指定のテーブルに対して <code>upload_insert</code> スクリプトが定義されていない場合において、<code>upload_new_row_insert</code> スクリプトを呼び出したときにエラーを発生させたアップロード挿入ローの総数を返します。 以前は、無視されたアップロード済み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、<code>handle_error</code> または <code>handle_odbc_error</code> が定義されており、1000 が返された場合、または指定のテーブルに対して <code>upload_update</code> スクリプトが定義されていない場合において、<code>upload_update</code> スクリプトを呼び出したときにエラーを発生させたアップロード更新ローの総数を返します。 強制的な競合モードでは、<code>handle_error</code> または <code>handle_odbc_error</code> が定義されており、1000 が返された場合、または <code>upload_new_row_insert</code> または <code>upload_old_row_insert</code> スクリプトを呼び出したときにエラーを発生させたアップロード更新ローの総数を返します。 以前は、無視されたアップロード済み更新の数を返していました。
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正常に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、統合データベースに正常に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。

統計プロパティ	説明
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正常に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、CHAR、VARCHAR、LONG VARCHAR、NCHAR、NVARCHAR、LONG NVARCHAR データ型のリモート・データベース・カラムに含まれる 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** スクリプトを登録するための Mobile Link システム・プロシージャでは、Oracle 統合データベース用にスクリプトの内容パラメータで CLOB データ型が使用されるようになりました。ml_add_property システム・プロシージャでは、Oracle 用に prop_value パラメータが CLOB 型になりました。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 を生成するようになりました。これは、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

きに生成されます。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、および 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

「[動作の変更と廃止予定機能](#)」 [244](#) ページを参照してください。

SQL Anywhere クライアント

● ダウンロード・エラー・フックは廃止予定 エラー・フック

`sp_hook_dbmsync_download_com_error`、`sp_hook_dbmsync_fatal_sql_error`、`sp_hook_dbmsync_sql_error` は廃止される予定です。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_dbmsync_log_rescan` は `dbmsync` 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_dbmsync_log_rescan` フックが呼び出されていました。そのため、`dbmsync` 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、コマンド・ラインで `dbmsync -n` オプションが複数回指定された場合や、スケジュールが有効になっている場合など、`dbmsync` が別の同期を予期した場合にかぎりこのフックが呼び出されるようになりました。

「[sp_hook_dbmsync_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ましたが、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 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・オプションを指定するだけで済みます。
[「Mobile Link クライアント・ネットワーク・プロトコル・オプションの一覧」](#) [『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 サーバで旧バージョンのデータベースを使用するには、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」](#) 392 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
<i>dbmlsv9.dll</i>	<i>mlodbc10.dll</i>
<i>dbasdesk.dll</i>	<i>mlasdesk.dll</i>
<i>dbasdev.dll</i>	<i>mlasdev.dll</i>
<i>dbmlsrv.mle</i>	<i>mlsrv10.mle</i>
<i>syncasa.sql</i>	<i>syncsa.sql</i>

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・アクセスで使用する ODBC の変更」 258 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 デバイスをサポートするための SDK は削除されました。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** dblsn -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。
- **サーバ起動同期における Kyocera デバイスのサポート終了** 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 デバイスをサポートします。
[「Palm デバイス用 Mobile Link Listener C API」](#) [『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 バージョン 10.0.0 の新機能、動作の変更、廃止予定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。 .NET または Java アプリケーションからモバイル Web サービスを使用し易くするために、WSDL コンパイラが用意されています。

「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、1つのストア・プロシージャによって、データベースへのローの挿入と、別のアプリケーションへのメッセージ送信を実行し、さらに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させることができます。

[「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・セレクト** 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・セレクトを作成する構文は、転送ルールの条件に使用する構文と同じです。

[「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・セレクト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。

[「セレクト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・プロパティの列挙体」](#) [『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もできます。

[「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 の起動」『QAnywhere』を参照してください。

- **フェールオーバー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、フェールオーバー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。またこのオプションで、Push 通知に永続的な接続を使用するかどうか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。

- 「`-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 がメッセージのタイムスタンプ設定に ISO 8601 日時フォーマットを使用** 情報、警告、エラーの各メッセージのタイムスタンプに、明確に定義された ISO 8601 日時フォーマット (`{[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、各クライアントに対して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。

- **メッセージ・プロパティの追加** 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 送信先を含めることができるようになりました。

「JMS コネクタへの QAnywhere メッセージの送信」 『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。

動作の変更と廃止予定機能

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、デフォルトでクライアント・メッセージ・ストア ID が指定されます。

Mobile Link ユーザ名を管理するために、新しい qaagent オプションが用意されています。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・アクセスで使用する ODBC の変更](#)」 258 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、または使用しない Push 通知を有効に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。このため、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・ファイルは廃止予定** プロパティは、ファイルではなくデータベースに格納され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。この関数は、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、多くの機能強化が加えられています。

SQL Remote

次の項では、SQL Remote バージョン 10.0.0 の新機能、動作の変更、廃止予定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 (dbremote)」 『SQL Remote』の -ux オプションを参照してください。
- **dbremote がメッセージのタイムスタンプ設定に ISO 8601 日時フォーマットを使用** 情報、警告、エラーの各メッセージのタイムスタンプに、明確に定義された ISO 8601 日時フォーマット ({||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** 完了時にウィンドウを閉じるには、新しい -qc オプションを使用してください。
「Message Agent (dbremote)」 『SQL Remote』を参照してください。

動作の変更と廃止予定機能

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 の統合データベースはサポートされなくなりました。このため、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に含まれなくなりました。
Adaptive Server Enterprise データベースを同期するには、Mobile Link 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** データベース抽出ウィザードは、バージョン 10 のデータベースだけに使用できます。
- **#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 \(dbxtract\)](#)」 [『SQL Remote』](#) を参照してください。

● **Message Agent (dbremote) の廃止予定のオプション** 完了時にウィンドウを閉じるための -k オプションは廃止される予定です。完了時にウィンドウを閉じるには、新しい -qc オプションを使用してください。

Ultra Light

次の項では、Ultra Light バージョン 10.0.0 の新機能、動作の変更、廃止予定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 の占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、「[Ultra Light 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されました。特に、テーブルのローの最大数は 1600 万にまで増加されています。現在のデータベースの上限に関する詳細については、「[Ultra Light 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 作成パラメータ](#)」『[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。
- **データベース・パフォーマンスとデータ整合性の向上** インデックス処理とデータベース・ページ管理が改善された結果、Ultra Light のデータベース・パフォーマンスとデータ整合性が全体的に向上しました。
- **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改善につながったり、データベース・ファイル・サイズに影響を及ぼしたりする可能性があります。「[Ultra Light のパフォーマンスと最適化](#)」『[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。Ultra Light データベースのソースとして、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で設定して、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」 419 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 Windows CE デバイスに展開されたデータベースに直接接続できます。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、Ultra Light データベースを指定できます。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 Mobile」](#) [『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。これを行うには、`ulrt.lib` ではなく `ulrtc.lib` に対してリンクします。

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」 419 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 11 - 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、引き続き既存のランタイムを使用できます。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](#)」『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は、デフォルトのユーザ ID **DBA** (パスワード **sql**) で作成されます。したがってユーザ ID、パスワード、信頼できるルート証明書は、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・プロパティ](#)』『[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](#)』『[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 ～ 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 max_hash_size 作成パラメータ](#)』『[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。
- **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。『[Ultra Light checksum_level 作成パラメータ](#)』『[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。
- **BLOB サポートの拡張** Ultra Light データベースでの 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 データベースの作成、変更、管理をグラフィカルなユーザ・インタフェースから実行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 には次のウィザードが含まれています。
 - 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提供します。
 - 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を持つユーティリティはありません。
 - データベース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提供します。
 - 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提供します。

- 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を持つ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** データベースの同期をとります。このウィザードは **ulsync** ユーティリティと同じ機能を提供します。
 - データベース・アップグレード・ウィザードを使用して、既存の **Ultra Light** データベースを以前のバージョンからアップグレードします。このウィザードは **ulload** ユーティリティを併用した場合の **ulunloadold** ユーティリティと同じ機能を提供します。
 - データベース・アンロード・ウィザードを使用して、**Ultra Light** データベースから XML、SQL、または別のデータベースにデータ/スキーマ情報をアンロードします。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を併用した場合の **ulunload** ユーティリティと同じ機能を提供します。
- **Interactive SQL** Interactive SQL を使用して、**Ultra Light** データベースで SQL 文の開発とテストを実行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 **ulsql** 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」](#) 『[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。
- **データベース作成パラメータの拡張** 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、拡張された作成パラメータを使用できるようになりました。これらの拡張された作成パラメータは、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機能を設定できます。「[Ultra Light で使用するデータベース作成パラメータの選択](#)」『[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](#)」『[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、ulsync では、パブリケーション・マスクだけではなく、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をカンマで区切ったリストで指定します。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 \(ulcond11\)](#)」『[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。
- **ulmbreg** Ultra Light for AppForge を登録するための ulmbreg 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ます。

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 を使用してテーブルを作成するだけでなく、この文を使用して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 つの文を使用した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 文を使用して、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「SQL 関数」『SQL Anywhere サーバ - SQL 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、同期速度に悪影響を与えていました。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 (ulcond11)」『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](#)」『[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。「[Additional Parameters 同期パラメータ](#)」『[Ultra Light データベース管理とリファレンス](#)』または次の項目のいずれかを参照してください。

- 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、接続文字列に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/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 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。このため、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 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](#) を呼び出す必要があります。「[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](#)』、「[ULEnableTcpiSynchronization 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](#)』、「[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](#) を使用すると、アプリケーションのサイズが大幅に減少します。「[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、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 は不要になりました。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になるなので、ULEnableUserAuthentication 関数は廃止される予定です。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 関数は同期中の zlib 圧縮を有効にします。
「ULEnableZlibSyncCompression 関数」『Ultra Light - C/C++ プログラミング』と「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。

注意

zlib 圧縮は、Palm OS または Symbian OS では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。関数が返す値は、スキーマの種類によって異なります。

- 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。
 - `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、`SQL_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。「[AdditionalParms 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` メソッドが認識するプロパティが増えました。「[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 プロパティが含まれ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1つのデータベースに同時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ための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。
 - 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** 既存のメソッドが次のように強化されました。
 - 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利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 11 - 紹介](#)』を参照してください。

動作の変更と廃止予定機能

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。

廃止予定のプラットフォーム

- 今回のリリースで、PocketPC 2000 OS のサポートは廃止される予定です。
- CodeWarrior 8 はサポートされなくなりました。代わりに CodeWarrior 9 を使用してください。
- Windows CE MIPS プロセッサはサポートされなくなりました。

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」 419 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」 419 ページ](#)を参照してください。

削除、廃止予定、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 ファイルがなくなったため、データベース・スキーマに接続したり、データベース・スキーマを動的にアップグレードすることはできなくなりました。このような従来の Ultra Light の機能に関連するすべてのクラスと関数は削除されました。
 - 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` はリリース 10 で非推奨になりました。`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は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` は廃止される予定です。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 `tcip`、`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」](#) [『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。
- `ULSocketStream`、`ULHTTPStream`、`ULHTTPSSStream` は、必要な文字列を適切に返すように変更されました。
- Windows CE デバイスのための `ULActiveSyncStream` は削除されました。Windows メッセージ・ハンドラで同期メッセージを受信する場合、`ActiveSync` プロバイダに登録されている `Ultra Light` アプリケーションでは、`ULActiveSyncStream` の代わりに標準の同期ストリームを使用してください。
- **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` ファイルがなくなったため、データベース・スキーマを動的にアップグレードできなくなりました。このような従来の `Ultra Light` の機能に関連するすべてのクラスと関数は削除されました。
- **Ultra Light.NET API** `Ultra Light.NET` API の関数は次のように変更されました。
 - `.usm` ファイルがなくなったため、データベース・スキーマに接続したり、データベース・スキーマを動的にアップグレードすることはできなくなりました。このような従来の `Ultra Light` の機能に関連するすべてのクラスとメソッドは削除されました。
 - `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` はこの列挙体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` ファイルがなくなったため、データベース・スキーマに接続したり、データベース・スキーマを動的にアップグレードすることはできなくなりました。このような従来の `Ultra Light` の機能に関連するすべてのクラスとメソッドは削除されました。
- **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` ファイルがなくなったため、データベース・スキーマに接続したり、データベース・スキーマを動的にアップグレードすることはできなくなりました。このような従来の Ultra Light の機能に関連するすべてのクラスとメソッドは削除されました。

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。
- **ulmbvreg** ulmbvreg 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` 下の `win32` ディレクトリにインストールされます。

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 \(ulcond11\)」](#) [『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、廃止予定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点や追加機能は、次の項で説明します。

- [「SQL Anywhere プラグイン」 248 ページ](#)
- [「Sybase Central の新しいプラグイン」 249 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。[「Sybase Central のナビゲーション」](#) [『SQL Anywhere サーバ - データベース管理』](#)を参照してください。
- **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] ウィンドウを開き、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] ウィンドウ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** Sybase Central で [表示] - [検索ウィンドウ枠] を選択することで、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、プラグイン間を簡単にナビゲーションできます。これは、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特に便利です。

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・バックアップからのリストア」](#) [『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** Sybase Central と Interactive SQL の [オプション] ウィンドウの [フォーマット] タブで、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」](#) 225 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 データベースの作成、変更、管理をグラフィカルなユーザ・インタフェースから実行できるようになりました。
[「グラフィカルな管理ツール」](#) 236 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 プラグインの従来の機能も強化され、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」](#) 201 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 文の開発とテストを実行できるようになりました。ulsql ユーティ

リティは廃止される予定です。「グラフィカルな管理ツール」 236 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 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 での 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 での @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、指定の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 での @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、指定の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。

動作の変更と廃止予定機能

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。
- **Interactive SQL から返される結果セット数に制限はない** 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 におけるグラフィカルなプランへのアクセスと保存**
 - 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ファイル拡張子 *.xml* 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、以前の *.xml* ファイル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、この拡張子も引き続きサポートされます。
 - [オプション] ウィンドウで、*.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 キーワードは廃止予定** 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る予定です。「[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 文によって現在の Interactive SQL ウィンドウが閉じられる** 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_MULTUSER の新しいエラー・コード** SQLE_ENGINE_NOT_MULTUSER を処理するようにプログラムされたアプリケーションでは、新しいエラー・コードのチェックを行う必要があります。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としたときに、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_MULTUSER 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 列挙体](#)」『[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。ユーティリティには Sybase Central の [ツール] メニューからもアクセスできます。

廃止予定機能とサポート終了機能

- **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、および Sybase Central 内の SQL Anywhere プラグインと Mobile Link プラグインに接続するための [接続] ウィンドウで、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 のデータベース・サーバと、バージョン 7 の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 の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、または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可能です。「[バージョン 9 以前のデータベースをバージョン 11 用に再構築](#)」399 ページを参照してください。
- **isql_log オプションは廃止予定** 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る予定です。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 の説明で示したファイルの変更点のほかに、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](#)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法に重点を置く新しい章が追加されました。
『[SQL Anywhere グラフィカル管理ツール](#)』『[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** 製品をインストールすると、該当のインストール・ビルドに関する説明を含んだ、サポートされているプラットフォームのページもインストールされるようになりました。これらのページは、SQL Anywhere インストール環境の *documentation* 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。マニュアルには、必要に応じてこれらのページへのリンクが含まれています。
- **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](#)』というマニュアルは削除されました。SQL Anywhere のセキュリティ機能については、[SQL Anywhere サーバ - 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** 『[Mobile Link の ODBC ドライバ](#)』というマニュアルは削除されました。

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、廃止予定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。ツールの完全なマニュアル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途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。

サポート・ユーティリティ (dbsupport) を使用して、好きなときに機能の統計値を送信することもできます。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。「[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ウィンドウが表示されます。

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ウィンドウが表示されます。これらのエラー・レポートが自動的に送信されるようにサポート・ユーティリティ (dbsupport) を設定すること

もできます。「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。

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。

- **管理ツールで利用できる機能の制御** 管理ツールでユーザが使用可能な機能を制御できるようになりました。これを行うには、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。TCP/IP を代わりに使用できます。

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点を示します。

- **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	<i>samples-dir¥demo.db</i>
Mobile Link CustDB サンプル統合データベース・アプリケーション	<i>samples-dir¥MobiLink¥CustDB¥</i>
Ultra Light CustDB サンプル・データベース	<i>samples-dir¥UltraLite¥CustDB¥</i>

サポートされているオペレーティング・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。

- **サポート対象外のプラットフォーム** 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点については、「[サポートされるプラットフォーム](#)」『[SQL Anywhere 11 - 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 11 - 紹介](#)』を参照してください。

- **その他のファイル名の変更** 製品別に示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
<i>asa.cvf</i>	<i>sqlany.cvf</i>
<i>asaldap.ini</i>	<i>saldap.ini</i>
<i>asasrv.ini</i>	<i>sasrv.ini</i>
<i>asa_config.sh</i>	<i>sa_config.sh</i>
<i>install-dir/SYBSasa9/lib</i>	<i>sqlanywhere10/lib32</i> または <i>sqlanywhere10/lib64</i>
<i>install-dir/SYBSasa9/bin</i>	<i>sqlanywhere10/bin32</i> または <i>sqlanywhere10/bin64</i>

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)

製品別に示したファイルの変更点のほかに、*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
Replication Agent	ASANYLTM	SQLANYLTM

Mobile Link、QAnywhere、リモート・データ・アクセスで使用する 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、別途ダウンロードできます。

http://www.iAnywhere.jp/developers/technotes/odbc_mobilink.html を参照してください。

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	260
バージョン 9.0.2 での動作の変更	277

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 [接続テスト] 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・サーバのプロパティ] ウィンドウにも [接続テスト] 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。
- **派生テーブルが実行プランに表示される** 派生テーブルが、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 [Web サービス]」 『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** Sybase Central の [データのアンロード] ウィンドウを使用して、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] ウィンドウを使用したデータのエクスポート」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

- **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させ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** ファイル難読化ユーティリティ (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。

「ファイル難読化ユーティリティ (dbfhide)」 『SQL Anywhere サーバ - データベース管理』を参照してください。

- **FIPS 認定セキュリティ** 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 認定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 認定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と復号化」 『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・サーバをシェル・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-Window がサポートされている Linux および 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] ウィンドウとデータベース・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れない場合は、サーバはシェル・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-Window がサポートされている Linux および 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] ウィンドウとデータベース・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 (旧式)」 『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 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 (旧式)」 『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」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 (9.0.2.1234 など) 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 認定セキュリティ・ストリーム** Windows デバイスで、Certicom の FIPS 140-2 認定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。
- **廃止予定のセキュリティ機能** 「Mobile Link の動作の変更」 279 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。

「[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す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。

「[Windows デバイス用の Listener ユーティリティ](#)」『[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

「[Windows 用の Listener アクション・コマンド](#)」『[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。

「[Palm デバイス用ユーティリティ](#)」『[Mobile Link - サーバ起動同期](#)』を参照してください。

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 \(dbremote\)](#)」 [『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廃止される予定です。

「[チュートリアル : Ultra Light.NET アプリケーションのビルド](#)」 [『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 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。

「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 (旧式)」『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。

「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 2.0 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、「**expanded mode**」というコード生成モードをサポートしています。このモードを使用すると、グローバル・データのメモリ利用効率が向上します。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。

- **信頼できる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、安全な同期の信頼できる証明書は、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、信頼できる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。
 - **実行プランの表示** 実行プランには、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 実行プランの記述の強化** Ultra Light Interactive SQL で表示できる Ultra Light 実行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実行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

- **フェールオーバー・サーバ** QAnywhere Agent 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。
[「フェールオーバー・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・ライブラリが、メッセージ・リスナをサポートす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。

「[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

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

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 Mobile」](#) 『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。
[「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](#) 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 デバイス用 Mobile Link Listener C API」](#) 『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廃止予定機能についても説明します。

廃止予定機能は変更される可能性があります

廃止予定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。

Adaptive Server Anywhere の動作の変更

廃止予定機能とサポート終了機能

次に、サポートを終了した機能、廃止予定の機能の中で、既存のアプリケーションに影響する可能性があるものを示します。

- **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。

「[カラム統計の更新によるオプティマイザのパフォーマンス向上](#)」『[SQL Anywhere サーバ - SQL の使用法](#)』を参照してください。

- **廃止予定のデータベース・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
- **廃止予定の古いデータベース・フォーマット** 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。
- **廃止予定の UNIX 用非スレッド型 DBTools ライブラリ** UNIX 用の非スレッド型 DBTools ライブラリ (libdbtool9.so) は廃止される予定です。このライブラリは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・オプションを使用した HTTP ポートおよび HTTPS ポートでの Web 要求の受信は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 DB 領域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 DB 領域ファイル名は、末尾に **R** が追加されます。これは、新しい DB 領域ファイルが、元の DB 領域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、DB 領域ファイル名には **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予定の 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る予定です。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。
『[Mobile Link Listener ユーティリティ](#)』 『[Mobile Link - サーバ起動同期](#)』の「**-I option**」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。

SQL Remote の動作の変更

廃止予定機能とサポート終了機能

次に、サポートを終了した機能、廃止予定の機能の中で、既存のアプリケーションに影響する可能性があるものを示します。

- **廃止予定の SQL Remote for Adaptive Server Enterprise** 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 \(dbxtract\)](#)』 『[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。

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、廃止予定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。

廃止予定機能およびサポート終了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。

廃止予定機能とサポート終了機能

次の機能は、廃止予定であるか、サポートが終了しました。

- **廃止予定の静的インタフェース** 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 `SQL_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。

QAnywhere の動作の変更

廃止予定機能とサポート終了機能

- **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 <i>id</i>	-id <i>id</i>
-dbauser <i>user</i>	-c "UID= <i>user</i> "
-dbeng <i>name</i>	-c "ENG= <i>name</i> "
-dbfile <i>filename</i>	-c "DBF= <i>filename</i> "
-dbname <i>name</i>	-c "DBN= <i>name</i> "
-ek <i>key</i>	-c "DBKEY= <i>key</i> "
-password <i>password</i>	-c "PWD= <i>password</i> "
-sv	-c "Start={ dbeng9 dbsrv9 }"
-verbose	-v[<i>levels</i>]

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 ユーティリティ](#)」 [『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。

「[JMS コネクタから QAnywhere クライアントへのメッセージの送信](#)」 [『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ーヘッドの縮小** 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。

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	286
バージョン 9.0.1 での動作の変更	301

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*。
- **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・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。

「[ParallelIndexScan 方式](#)」 『[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。

「[ClusteredHashGroupBy アルゴリズム \(GrByHClust\)](#)」 『[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のインデックス](#)」 『[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、廃止される予定です。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**

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] ウィンドウによって、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。

● **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』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] ウィンドウによって、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。
- **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 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。

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。
- **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、現在のユーザの外部ログインを作成できるようになったため、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] ウィンドウによって、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。
- **Sybase Central の [表示] メニューを使用してカラムをソート可能** Sybase Central の [表示] 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。
- **[外部キーのプロパティ] ウィンドウから外部キー設定を変更可能** 外部キー設定は、Sybase Central の [外部キーのプロパティ] ウィンドウから変更できます。
- **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・ウィンドウがあります。
- **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 [サービスのプロパティ] ウィンドウでは、サービスの作成時と編集時に **domain\user** という形式で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、[サービスのプロパティ] ウィンドウの [アカウント] タブやサービス作成ウィザードの、[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、データベース・サーバ・メッセージ・ウィンドウからのメッセージがロギングされるファイル名を返します。
「データベース・サーバ・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・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 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・サーバ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 サーバのシステム・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](#) 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。

- **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、これらのクラス名は `SyncDB2_901.class` および `SyncDB2Long_901.class` です。

「[IBM DB2 LUW 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に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 \(旧式\)](#)」 『[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・ゲートウェイ」『[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。
「[Sybase Central を使用したサーバ側設定の実行](#)」『[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。
「[Windows デバイス用の 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。
- **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。
「[Mobile Link 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 デバイス用の Listener 設定ユーティリティ](#)」『[Mobile Link - サーバ起動同期](#)』を参照してください。

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] ウィンドウからしか使用できなかった情報を含む新しいカラムや、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** Mobile Link モニタ・テーブルのソート機能が改善されました。データが追加または更新され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** チャート内でズーム・イン、ズーム・アウト、または選択範囲に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 (旧式)」 『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ビルド」 『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。サブクエリは、FROM 句で派生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 ステーションナリ** ステーションナリ **Palm OS UltraLite C++ Component App** 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。

「[Palm OS 用 Ultra Light 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。

- **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 \(旧式\)](#)」 『[Ultra Light - C/C++ プログラミング](#)』と「[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え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](#)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](#)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 \(旧式\)](#)」 『[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](#)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](#)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

廃止予定機能とサポート終了機能

次に、サポートを終了した機能、廃止予定の機能の中で、既存のアプリケーションに影響する可能性があるものを示します。

- **サポート終了した MDSR 暗号化** 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。この変更により、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** オプションの指定が不要になりました。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・ファイルは廃止予定** 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・ファイルは廃止予定** 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 オプションは廃止予定** 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 オプションは廃止予定** 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す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] ウィンドウ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させるために、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する [接続] ウィンドウの [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できないようになりました。

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] ウィンドウと [ODBC の設定] ウィンドウの [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、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 ～ 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** Embedded 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) 関数でステータスや同期情報を管理する必要がなくなったため、これ

らの関数は廃止される予定です。また、ULData と ULConnection Reopen メソッドも廃止される予定です。

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 アプリケーションへの HotSync 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。

- **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。
- **UEnableGenericSchema 関数は廃止予定** 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、UEnableGenericSchema 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。

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	306
バージョン 9.0 での動作の変更	325

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。必要に応じて、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 句」 『SQL Anywhere サーバ - SQL リファレンス』
- 「INTERSECT 句」 『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。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があります。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 句の中でラテラル派生テーブルによる外部参照が可能** FROM 句の中で、派生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ー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 [Web サービス]」 『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。

「データベース・サーバ・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、シングルスバイト対シングルス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](#)』、
「[データベース・パフォーマンスの改善](#)」『[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、「データベース・サーバ・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) を使用する場合は、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 ～ 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」 『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。

「SQL 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 の動作の変更」 325 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。
- **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。

「データベース・サーバ・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。
「SortTopN アルゴリズム (SrtN)」 『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ています。
「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実行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。
- **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。
- **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。

- **optimistic_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。

- **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任意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」 『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・ウィンドウ、ウィンドウ、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 (データベース・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 の使用方法』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・オプションの使用」『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] ウィンドウを使用すると、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。
- **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 のナビゲーション」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **グラフィカルなプランの強化** グラフィカルなプランの表示が、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] ウィンドウの [結果] タブで設定します。

- **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] ウィンドウの強化** Interactive SQL の [コマンド履歴] ウィンドウでは、コマンドのコピーや削除、およびウィンドウ内での複数コマンドの選択を実行できます。コマンド履歴は、Interactive SQL セッション間で保持されます。
「[SQL 文、実行プラン、結果セットの印刷](#)」 『[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。
「[Mobile Link 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。

Mobile 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空白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、後続空白が削除されます。

[「-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 (日) 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 ワーカー・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 UltraLite 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。

- **UltraLite 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。
[「Ultra Light for C/C++ の開発者」](#) [『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 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。
- **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。
- **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・フェーズでは、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 Unicode ライブラリ** Embedded SQL と静的型 C++ API アプリケーション用に、Unicode 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。
「SQL Anywhere での Java サポート」『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の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 '%I' はあいまいです。」 [『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のため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す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。

「[SortTopN アルゴリズム \(SrtN\)](#)」『[SQL Anywhere サーバ - SQL の使用法](#)』を参照してください。

廃止予定機能とサポート終了機能

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](#)」 325 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 -esc 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_sync_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 Acknowledgement 同期パラメータ」 『Ultra Light データベース管理とリファレンス』

- **Windows CE デバイス上でデフォルトでは動作しない dbmsync フックがある** dbmsync 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。

dbmsync のイベント・フックである `sp_hook_dbmsync_download_com_error`、`sp_hook_dbmsync_download_fatal_sql_error`、`sp_hook_dbmsync_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_dbmsync_download_com_error (旧式)」 『Mobile Link - クライアント管理』
- 「sp_hook_dbmsync_download_fatal_sql_error (旧式)」 『Mobile Link - クライアント管理』
- 「sp_hook_dbmsync_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。dbmsync の場合、エラーは GUI と出力ファイルに出力されます。Ultra 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」 『エラー・メッセージ』を参照してください。

- **廃止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。

- **廃止予定の -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。
- **廃止予定の mlxtract** `mlxtract` ユーティリティは廃止される予定です。

「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かったことがありました。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 ファミリー・メンバーは、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ことから、「静的インタフェース」と呼ばれるようになりました。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 Unicode 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。

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	334
バージョン 8.0.2 での動作の変更	344

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の解釈](#)」『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の解釈](#)」『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。

- **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。

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せん。

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 にはファイル難読化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、単純暗号化によって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、「[ファイル難読化ユーティリティ \(dbfhide\)](#)」『[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。現在では、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 Mobile 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] ウィンドウでは、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。
- **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 システム・プロシージャ](#)」『[Mobile Link - サーバ管理](#)』、および「[ml_add_dnet_table_script 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。
- **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](#)」『[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。

- **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 \(旧式\)](#)」『[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。

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させるために、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、「[レプリケーション・エラーのレポートと処理](#)」『[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・テクノロジーを、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 テクノロジーを提供します。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](#)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 テクノロジーが使用されます。Certicom テクノロジーを使用するには、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 11 のコンポーネント](#)」 『[SQL Anywhere 11 - 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](#)」『[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。

- **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、情報の多い同期ウィンドウを設計できます。

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**
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、「[オプション '%1' の設定が無効です。](#)」『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_size	[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。

廃止予定機能とサポート終了機能

次に、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する可能性があるものを示します。

- **Windows で廃止予定の -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.net/ssl-certificate-services/managed.htm> を参照してください。

- **dbmlsrv の廃止予定オプション -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・オプションは廃止される予定です。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 はサポート対象外** 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。

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	350
バージョン 8.0.1 での動作の変更	356

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** データベース・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。

- **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るならば、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・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、「[データベース・サーバ・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 \[Web サービス\]](#)」『[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.getType 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、便利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] ウィンドウには、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。新しい **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、「[Additional Parameters 同期パラメータ](#)」『[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。

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。

廃止予定機能とサポート終了機能

次に、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する可能性があるものを示します。

- **廃止予定の DEBUG 接続パラメータ** DEBUG 接続パラメータは廃止される予定です。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。
- **廃止予定の AGENT 接続パラメータ** AGENT 接続パラメータは廃止される予定です。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。
- **GLOBAL AUTOINCREMENT** 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。
- **GLOBAL AUTOINCREMENT** 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、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、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。

廃止予定機能とサポート終了機能

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 までは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。

バージョン 8.0.0 の新機能

目次

バージョン 8 の新機能	360
バージョン 8 での動作の変更	382

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 11 へのアップグレード](#)」 391 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」 360 ページ
- 「セキュリティ」 362 ページ
- 「SQL の機能」 363 ページ
- 「開発ツールと管理ツール」 364 ページ
- 「アプリケーション開発」 366 ページ
- 「管理機能とトラブルシューティング」 367 ページ
- 「クライアント／サーバ接続」 370 ページ
- 「データベース内の Java」 371 ページ
- 「マニュアル」 372 ページ
- 「その他」 372 ページ

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・ログの概要](#)」『[SQL Anywhere サーバ - データベース管理](#)』を参照してください。

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](#)」『[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 11 のコンポーネント](#)」『SQL Anywhere 11 - 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。

- **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。

- **[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、-d1 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](#)」『[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、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』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とスキューの削減](#)」『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、データベース・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** 接続デバッグ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: プレフィックスが削除され、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・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、「データベース・サーバ・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 デバッグ情報をデータベース・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ー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、システムで利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 \(旧式\)](#)」『[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、こ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できるようになり、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。

- **冗長ログイン** 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 サーバのログインを設定できます。

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](#)」『[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。

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 データベースの保護](#)」 [『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](#)」『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 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` 構造体の `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](#)」『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設計されているので、通常の状態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・サイズを選択** 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** Ultra Light plug-in for CodeWarrior 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。
- **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ある。
- ストリーム・パラメータの指定が不要。

ULPalmDBStream 関数と **ULConduitStream** 関数は廃止される予定です。

詳細については、「[Palm アプリケーションへの HotSync 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。

- **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。
- **実行プラン情報** 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 に使用される実行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** Embedded 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 システム・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。

新たに廃止予定の機能およびサポートが終了した機能のリストについては、「[廃止予定機能とサポート終了機能](#)」 [386 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 11 のコンポーネント](#)」『[SQL Anywhere 11 - 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ため、`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 はサポート対象外** 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。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す。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ることがあります。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 \[データベース\]](#)」『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 環境変数を見捨てる** 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 環境変数を見捨て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。

廃止予定機能とサポート終了機能

次に、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する可能性があるものを示します。

- **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。
- **廃止予定の照合** 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。

廃止された照合	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、サポートされなくなりました。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。
- **廃止予定の None パラメータ** `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。
- **廃止予定の WITH HASH SIZE n 句** `WITH HASH SIZE` 句は、サポートされなくなりました。
- **廃止予定の max_work_table_hash_size オプション** `max_work_table_hash_size` オプションは、サポートされなくなりました。
- **廃止予定の max_hash_size オプション** `max_hash_size` オプションは、サポートされなくなりました。
- **廃止予定の SATMP 環境変数** 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。
- **廃止予定の dbmlsync StreamCompression 拡張オプション** 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。
- **廃止予定の ULPalmDBStream と ULConduitStream** 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。

SQL Anywhere 11 へのアップグレード

目次

SQL Anywhere のアップグレード	392
Mobile Link のアップグレード	410
QAnywhere のアップグレード	418
Ultra Light のアップグレード	419
SQL Remote のアップグレード	429

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 11 - 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 以降のデータベースのアップグレード

バージョン 10 以降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 11 のソフトウェアでバージョン 10 のデータベースを使用できるので、アップグレードや再構築を実行するかどうかは任意です。ただし、バージョン 11 の新機能をすべて利用したい場合は、データベースを再構築する必要があります。「[バージョン 10 以降のデータベースのアップグレード](#)」 [405 ページ](#)と「[バージョン 10 以降のデータベースの再構築](#)」 [396 ページ](#)を参照してください。

マテリアライズド・ビュー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、データベース内のマテリアライズド・ビューを再表示することをおすすめします。「[手動ビューのリフレッシュ](#)」『[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 11 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 11 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開始しようとすると、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 4 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 11 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](#)」 [402 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。使用しているアンロード・ユーティリティのバージョンを確認するには、**dbunload -?** コマンドを実行します。「[ユーティリティの使用](#)」 394 ページを参照してください。

また、バージョン 10 以降のデータベースで dbunload を使用する場合は、使用する dbunload のバージョンが、データベースへのアクセスに使用するデータベース・サーバのバージョンと一致する必要があります。dbunload のバージョンがデータベース・サーバのバージョンより古いまたは新しい場合は、エラーがレポートされます。

- 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](#)」 401 ページを参照してください。
- バージョン 11 のアンロード・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から、バージョン 11 のデータベース・サーバにある *reload.sql* ファイル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。スキーマの変更後は、新しいデータベースを初期化して、再ロード・スクリプトを適用できます。
- 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 11 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。廃止予定またはサポート対象外のデータベース・オプション設定、オブジェクト、または SQL 構文が *reload.sql* ファイルにアンロードされる可能性があるため、この方法は他の方法が失敗する場合にのみ使用してください。この方法を使用して再ロード中に問題が発生した場合は、ファイルを手動で編集する必要があります。バージョン 11 以降では、内部再ロード機能により、このような問題の多くが解決されています。

Mac OS X のデータベースの再構築

Mac OS X 用の SQL Anywhere 9.0.2 は PPC でサポートされ、SQL Anywhere 10.0.0 以降は Intel でサポートされていました。Mac OS X に 9.0.2 以前のバージョンのデータベースがある場合は、次の 2 つの方法でデータベースをアンロードできます。

- バージョン 9.0.2 のソフトウェアを使用してデータベースをアンロードします。
- SQL Anywhere 11 がインストールされている別のプラットフォームにデータベースをコピーしてから、バージョン 11 のソフトウェアを使用してデータベースをアンロードします。

データベースをアンロードした後は、Mac OS X でバージョン 11 のソフトウェアを使用してデータベースを再ロードできます。

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 11 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 11 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 11 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 Unicode 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 Unicode 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・ファイルのロケーションを指定するために使用されていた検索順序は、バージョン 11 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 11 のデータベース・サーバに接続する場合、SATMP (バージョン 11) 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 では、次のように旧バージョンのデータベースとデータベース・サーバを管理できます。
 - バージョン 9 以降のデータベース・サーバで動作するバージョン 9 以降のデータベースに接続して、管理できます。
 - Sybase Central と同じコンピュータで実行されているバージョン 5 以降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。データベースはアンロードの前に停止されます。
 - 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 7 以前のデータベースはサポートされません。

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 11 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 11 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 11 の実行プログラム・ディレクトリが、旧バージョンの実行プログラム・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 11 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。

アップグレードを行う前の重要な注意事項

SQL Anywhere をアップグレードする前に、次の作業を行ってください。

- **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、既存のアプリケーションを更新してください。[「バージョン 11.0.1 の新機能」 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 11 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」 394 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** アップグレードを開始する前に、データベースを検証し、ソフトウェアとデータベースをバックアップします。後でリカバリできるように、アップグレードの終了時にデータベースのバックアップを完了します。
- **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。

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 11 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
```

詳細については、「バックアップのクイック・スタート」『SQL Anywhere サーバ - データベース管理』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。
3. バージョン 11 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 11 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 mydb11.db
```

5. 新しいデータベースは、バックアップしてから使用します。次に例を示します。

```
dbbackup -c "DBF=mydb11.db;UID=DBA;PWD=sql" new-db-backup-dir
```

6. 新しいデータベースは、検証してから使用します。次に例を示します。

```
dbvalid -c "DBF=mydb11.db;UID=DBA;PWD=sql"
```

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」 402 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」 401 ページ

バージョン 10 以降のデータベースの再構築

データベースを再構築するには、データベースをアンロードしてから再ロードすることでファイル・フォーマットをアップグレードします。ファイル・フォーマットをアップグレードすると、ディスク上でデータの保管とアクセスに使用されるフォーマットが変更され、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。

dbunload 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・データベースを再構築する場合は、**-ar** オプションまたは **-an** オプションを必ず使用してください。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。

バージョン 10 以降のデータベースで dbunload を使用する場合は、使用する dbunload のバージョンが、データベースへのアクセスに使用するデータベース・サーバのバージョンと一致する必要があります。dbunload のバージョンがデータベース・サーバのバージョンより古いまたは新しい場合は、エラーがレポートされます。

SQL Anywhere のインデックスが変更された結果、データベースをアンロードしてから再ロードして再構築すると、再構築されたデータベースが元のデータベースよりも小さくなる場合があります。このデータベースのサイズの縮小は、問題や、データが失われたことを示すものではありません。

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。

オートインクリメント・カラムのあるテーブルの再ロード

dbunload の **-I** オプションを指定すると、再構築されたデータベースのオートインクリメント・カラムに次に使用可能な値を保持できます。このオプションを指定すると、オートインクリメント値が含まれるテーブルごとに、生成される *reload.sql* スクリプトに `sa_reset_identity` システム・プロシージャへの呼び出しが追加され、SYSTABCOL.max_identity の現在の値が維持されます。

データベースの再構築

◆ データベースを再構築するには、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」 395 ページ](#)を参照してください。
2. [スタート] - [プログラム] - [SQL Anywhere 11] - [Sybase Central] を選択します。
3. アップグレードするデータベースを実行するバージョン 11 のデータベース・サーバを起動してから、Sybase Central でデータベースに接続します。
4. [ツール] - [SQL Anywhere 11] - [データベースのアンロード] を選択します。
5. データベース・アンロード・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. [現在のバージョンのサーバで実行中のデータベースをアンロードする] を選択し、リストからデータベースを選択します。[次へ] をクリックします。
7. [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。
8. データベースの新しいファイル名を指定します。
9. 新しいデータベースのページ・サイズを指定することもできますが、指定するページ・サイズは、データベース・サーバのページ・サイズ以下である必要があります。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。強

力な暗号化を選択した場合は、データベースを起動するたびに暗号化キーが必要です。[\[次へ\]](#) をクリックします。

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と復号化](#)」
『[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・ウィザードを使用したデータのエクスポート](#)」 『[SQL Anywhere サーバ - SQL の使用法](#)』を参照してください。

◆ **データベースを再構築するには、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](#)」 [395 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 11 のユーティリティのパスがその他のユーティリティのパス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](#)」 [394 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、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、再構築するデータベースの DBA 権限を持っている必要があります。

このコマンドにより、新しいデータベースが作成されます。既存のデータベースをアップグレード後のデータベースに置き換えるには、-an オプションの代わりに -ar オプションを指定します。-ar オプションを使用するには、パーソナル・データベース・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・データベース・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。「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。

バージョン 9 以前のデータベースをバージョン 11 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 11 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 Mobile データベースのアップグレードについては、「Windows Mobile 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 用の SQL Anywhere 9.0.2 は PPC でサポートされ、SQL Anywhere 10.0.0 以降は Intel でサポートされていました。Mac OS X に 9.0.2 以前のバージョンのデータベースがある場合は、次の 2 つの方法でデータベースをアンロードできます。

- バージョン 9.0.2 のソフトウェアを使用してデータベースをアンロードします。
- SQL Anywhere 11 がインストールされている別のプラットフォームにデータベースをコピーしてから、バージョン 11 のソフトウェアを使用してデータベースをアンロードします。

データベースをアンロードした後は、Mac OS X でバージョン 11 のソフトウェアを使用してデータベースを再ロードできます。

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

バージョン 11 のツールを使用してバージョン 9 以前のデータベースを再構築するときには、次の制限事項があります。

- 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 11 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、処理を続行できない場合は、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 の各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] ウィンドウの [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への直接ファイル・システム・アクセスを使用して、コンピュータで dbunload を実行します (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。

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 11 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。

- パスワードを最初に入力したのが大文字と小文字を区別しないデータベースだった場合、そのパスワードの大文字と小文字は区別されません。
 - パスワードを最初に入力したのが大文字と小文字を区別するデータベースだった場合、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 11 データベースのデフォルトのデータベース・ページ・サイズは 4096 バイトです。バージョン 11 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 オプションまたは dbunload の -ap 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』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。
 - **照合** バージョン 9 以前の SQL Anywhere では、CHAR データ型で使用する 1 つの照合しかサポートしていませんでした。この照合は、SQL Anywhere 照合アルゴリズム (SACA) を使用していました。バージョン 10 以降の SQL Anywhere では、照合アルゴリズムとして、SACA と UCA (Unicode 照合アルゴリズム) の 2 つをサポートしています。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 SACA 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、その照合は保持されます (内部アンロード)。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。「[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。

- **データベースのファイル・サイズ** SQL Anywhere のインデックスが変更された結果、データベースをアンロードしてから再ロードして再構築すると、再構築されたデータベースが元の

データベースよりも小さくなる場合があります。このデータベースのサイズの縮小は、問題や、データが失われたことを示すものではありません。

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。このウィザードを使用すると、データベースを再ロード・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 11 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、アンロードを続行する前にデータベースが停止します。

◆ データベースを再構築するには、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](#)」 395 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。
3. 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。
4. [スタート] - [プログラム] - [SQL Anywhere 11] - [Sybase Central] を選択します。
5. [ツール] - [SQL Anywhere 11] - [データベースのアンロード] を選択します。
6. データベース・アンロード・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。
7. [旧バージョンのサーバで実行中のデータベース、または実行中ではないデータベースをアンロードする] を選択します。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。
8. [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。
9. 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 11 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。強力な暗号化を選択した場合は、データベースを起動するたびに暗号化キーが必要です。[「データベースの暗号化と復号化」](#) [『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。[次へ] をクリックします。
11. 再構築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。
12. [完了] をクリックします。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。

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、元のデータベースをそのまま残して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 11 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。

◆ データベースを再構築するには、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」 395 ページ](#)を参照してください。
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 11 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」 394 ページ](#)を参照してください。
3. バージョン 11 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. 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。
5. 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
```

[「バックアップのクイック・スタート」](#) [『SQL Anywhere サーバ - データベース管理』](#)を参照してください。

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](#)」 394 ページを参照してください。

6. `-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" -an mydb11.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再構築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・データベース・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・データベース・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。

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 11 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ッ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 文で、トリガが定義されたテーブルの所有者名が指定されておらず、ユーザが <i>reload.sql</i> 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、CREATE TRIGGER 文は失敗し、テーブルが見つからなかったことを示すエラーが返されます。	テーブル名のプレフィッ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。 <pre>CREATE PROCEDURE p() BEGIN DECLARE NCHAR INT; SET NCHAR = 1; END;</pre>	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 @ を名前のプレフィッ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。
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、<i>reload.sql</i> ファイルに <code>ml_add_property</code> 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。新しいバージョン 11 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 11 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」 394 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。 <pre>SET TEMPORARY OPTION tsql_outer_joins='on';</pre> 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。

バージョン 10 以降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 11 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 以降のデータベースの再構築](#)」 396 ページを参照してください。

データベース・アップグレード・ウィザードでは、バージョン 9.0.2 以前のデータベースをバージョン 11 にアップグレードできません。既存のデータベースをバージョン 11 にアップグレードするには、dbunload またはデータベース・アンロード・ウィザードを使用してデータベースをアンロードし、再ロードする必要があります。「[バージョン 9 以前のデータベースのアップグレード](#)」 392 ページを参照してください。

警告

必ずデータベース・ファイルをバックアップしてからアップグレードしてください。既存のファイルにアップグレードを適用した場合、アップグレードに失敗すると、これらのファイルは使用できなくなります。データベースのバックアップの詳細については、「[バックアップとデータ・リカバリ](#)」 『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](#)」 395 ページを参照してください。
2. [スタート] - [プログラム] - [SQL Anywhere 11] - [Sybase Central] を選択します。
3. SQL Anywhere のプラグインから、アップグレードするデータベースに接続します。データベースはバージョン 11 のデータベース・サーバで実行されている必要があります。
4. [ツール] - [SQL Anywhere 11] - [データベースのアップグレード] を選択します。
5. データベース・アップグレード・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・ログをコピーしてアーカイブすることをウィザードで選択しなかった場合は、その操作を行います。

ヒント

データベース・アップグレード・ウィザードは、次の方法でもアクセスできます。

- データベースを右クリックし、[データベースのアップグレード] を選択します。
- データベースを選択し、[ファイル] - [データベースのアップグレード] を選択します。

◆ 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 (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](#)」 [395 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 11 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](#)」 [394 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、アップグレードするデータベースの DBA 権限を持っている必要があります。

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。

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」 407 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」 407 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」 408 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・リリースを使用する必要があります。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 コマンドを発行して、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. データベース・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ー中にのみ生じます。

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ーの起動」 『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。最初の方法の方が簡単ですが、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 以降のデータベースのアップグレード」 405 ページまたは「バージョン 10 以降のデータベースの再構築」 396 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. 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 以降のデータベースのアップグレード」 405 ページまたは「バージョン 10 以降のデータベースの再構築」 396 ページを参照してください。
3. プライマリ・サーバとミラー・サーバの両方を停止します。
4. 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。
5. dbtran ユーティリティを使用して、手順 4 で保存したトランザクション・ログを変換します。
このトランザクション・ログには、手順 1 でバックアップした以降、データベースに適用されたすべての変更が含まれます。

6. ローカル・データベース・サーバを使用して、再構築したデータベースを開始します。
7. Interactive SQL から READ 文を使用して、変換されたトランザクション・ログを適用します。
8. 再構築したデータベースを停止します。
9. アップグレードまたは再構築したデータベースとそ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。

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 11.0.0 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 11 の Mobile Link サーバは、バージョン 8、9、10、11 のクライアントで使用できます。バージョン 10 または 11 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、既存のアプリケーションを更新してください。[SQL Anywhere 11 - 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」 411 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」 415 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。

Mobile Link クライアントがバージョン 8 以降の場合、アップグレードは段階的に実行できます。

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」 415 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 9 以前の Ultra Light アプリケーション・コードをバージョン 11.0.0 に移植する」 424 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。

- [「動作の変更と廃止予定機能」 214 ページ](#)
- [「アップグレードを行う前の重要な注意事項」 395 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。

説明

- 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 以降のアップグレード

◆ 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 (SQL Anywhere 10.0.0 以降の場合)。

1. SQL Anywhere データベースをアップグレードします。
「バージョン 10 以降のデータベースのアップグレード」 392 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_sa.sql` という名前です。これは、SQL Anywhere インストール環境の `MobiLink¥upgrade¥version` にあります。`version` は、アップグレード元の SQL Anywhere バージョンを示します。

たとえば、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。

```
READ "c:¥Program Files¥SQL Anywhere 11¥MobiLink¥upgrade¥10.0.1¥upgrade_sa.sql"
```

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。テーブルの変更パーミッションを持つユーザ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、SQL 文 `SETUSER` 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、`ALTER USER` 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。

```
ALTER USER TO dbo IDENTIFIED BY "";
```

- 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。この手順は、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 11 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 11 にアップグレードします。

手順については、次の項を参照してください。

- 「バージョン 10 以降のデータベースのアップグレード」 405 ページ
- 「バージョン 10 以降のデータベースの再構築」 396 ページ
- 「バージョン 9 以前のデータベースをバージョン 11 用に再構築」 399 ページ

- 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。
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* という名前です。これは、SQL Anywhere インストール環境の *MobiLink* *Upgrade* *version* にあります。*version* は、アップグレード元の 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 SQL 文 *SETUSER* 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 11\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_server to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_active_remote_id to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_passthrough_script to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_passthrough to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_passthrough_status to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_passthrough_repair to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_repository_archive to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_delivery_archive to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_status_history_archive to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_repository_props_archive 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;  
GRANT EXECUTE ON dbo.ml_server_update to my_user;  
GRANT EXECUTE ON dbo.ml_server_delete to my_user;  
GRANT EXECUTE ON dbo.ml_add_passthrough_script to my_user;  
GRANT EXECUTE ON dbo.ml_add_passthrough to my_user;  
GRANT EXECUTE ON dbo.ml_add_passthrough_repair to my_user;  
GRANT EXECUTE ON dbo.ml_delete_passthrough_script to my_user;  
GRANT EXECUTE ON dbo.ml_delete_passthrough to my_user;  
GRANT EXECUTE ON dbo.ml_delete_passthrough_repair to my_user;
```

Adaptive Server Enterprise、Oracle、または Microsoft SQL Server の Mobile Link システム・テーブルのアップグレード

Adaptive Server Enterprise、Oracle、または Microsoft SQL Server の統合データベース内の Mobile Link システム・オブジェクトをアップグレードする必要があるのは、11.0.0 より前のバージョンの Mobile Link サーバを使用している場合だけです。

◆ 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 の場合)。

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" パーミッション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。これは、SQL Anywhere インストール環境の *MobiLink* %upgrade %version にあります。version は、アップグレード元の Mobile Link バージョンを示します。

たとえば、Microsoft SQL Server バージョン 9.0.2 の統合データベースをアップグレードするには、次のコマンドを実行します。

```
isql -S server_name -U user_name -P password -l
"c:¥Program Files¥SQL Anywhere 11¥MobiLink¥upgrade¥9.0.2¥upgrade_mss.sql"
```

IBM DB2 LUW のアップグレード

DB2 LUW の統合データベースをアップグレードする必要があるのは、11.0.0 より前のバージョンの Mobile Link サーバを使用している場合だけです。

◆ 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 LUW の場合)。

1. 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¥syncdb2.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 LUW 設定スクリプトを実行する方法については、「[IBM DB2 LUW 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. DB2 LUW 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_db2.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、アップグレード元の Mobile Link のバージョンを示します。

3. *upgrade_db2.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 11 のリモートを使用している場合は、バージョン 10.0.0 からバージョン 11.0.0 にアップグレードする必要があります。

バージョン 11 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 11 - 変更点とアップグレード 1 ページ](#)を参照してください。

バージョン 11 の Mobile Link サーバでサポートされるのは、バージョン 8、9、10、および 11 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。

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

Mobile 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 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。

バージョン 11 の Mobile Link クライアントでは、同期を実行するためにバージョン 11 の Mobile Link サーバが必要です。バージョン 11 の Mobile Link クライアントは、バージョン 11 より前の Mobile Link 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](#)」 410 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手順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](#)」 392 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. すべてのデータベース・アクティビティを停止します。
2. 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。
3. 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。
4. 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。
5. 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 11 用に再構築](#)」 399 ページを参照してください。
6. 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。

7. 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 11 用に再構築」 399 ページを参照してください。

8. 新しいデータベースを停止します。
9. 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。
10. 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、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。

11. **dbmlsync** を起動し、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。

12. 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・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、一度にではなく、段階的にリモート・データベースをアップグレードすることがより簡単になります。

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。また、クライアントをアップグレードして名前付きパラメータの使用を開始することで、スクリプト・パラメータを任意の順序で配置できます。

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](#)」 411 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 11 - 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

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、新機能と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。次の項を参照してください。

- 「Ultra Light の新機能」 81 ページ
- 「Ultra Light の動作の変更と廃止予定機能」 86 ページ

Ultra Light 11.0.0 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、システム・パスに注意して、使用している Ultra Light ユーティリティがバージョン 11 であることを確認してください。「[ユーティリティの使用](#)」 394 ページを参照してください。

バージョン 10 以降のデータベースのアップグレード

バージョン 10.0.0 以降の Ultra Light データベースでは、バージョン 11 のアプリケーション、またはバージョン 11 の Ultra Light ツールからデータベースに接続した際、データベースのファイル・フォーマットが自動的にバージョン 11 にアップグレードされます。データベースがアップグレードされたら、バージョン 10 のアプリケーション、ユーティリティ、ソフトウェアに接続できなくなります。「[バージョン 10.0.0 以降の Ultra Light データベースのアップグレード](#)」 420 ページ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

バージョン 9 以前の Ultra Light データベースでは、データベースを再構築してください。データベースを再構築するには、古いデータベースをアンロードしてから、バージョン 11 の新しいデータベースに再ロードします。Ultra Light 11 でバージョン 9 以前のデータベースに接続しようとすると、データベースの開始時にエラーが発生します。「[バージョン 9 以前の Ultra Light データベースのアップグレード](#)」 420 ページを参照してください。

既存のソフトウェアとの互換性

- バージョン 11 では、Ultra Light for AppForge はサポートされていません。
- Ultra Light 11.0.0 のデータベース・ファイルは、バージョン 11.0.0 のクライアント・アプリケーション、またはバージョン 11 の Ultra Light エンジンからの接続のみをサポートします。
- Ultra Light 11.0.0 は、バージョン 10 のデータベース・ファイルを自動的にアップグレードします。
- Ultra Light バージョン 11 ランタイムと Ultra Light バージョン 11 エンジンは、バージョン 9 以前の 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。
- 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 11 のデータベースは完全に管理できます。

- バージョン 10 のデータベースは完全に管理できます。バージョン 10 のデータベースは、自動的にバージョン 11 にアップグレードされます。
- バージョン 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。

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 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 用に初期バージョンの Palm を最適化する方法については、「[Ultra Light 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。

バージョン 10.0.0 以降の Ultra Light データベースのアップグレード

バージョン 10 の Ultra Light データベースを使用している場合は、バージョン 11 の Ultra Light アプリケーション、またはバージョン 11 の Ultra Light ツール (ulsync、ulunload、ulinfo、Sybase Central など) から接続した際に、自動的にバージョン 11 のデータベースにアップグレードされます。データベースをアップグレードすると、バージョン 10 のアプリケーションからはそのデータベースに接続できなくなります。

バージョン 9 以前の 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場合は、次の作業が必要です。

- 重要な情報が保存されているデータベースをアップグレードする場合は、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・ファイルをデスクトップ・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するまでは、*ulunloadold.exe* 以外の Ultra Light 11.0.0 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、別個の *.usm* ファイルに保存されるのではなく、データベースの一部になりました。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、Unicode 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 作成パラメータ](#)」『[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。

Windows Mobile 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、Unicode 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 コード化すると、データベースのサイズを不必要に増加させます。

- バージョン 10 では、データベースでの設定にかかわらず、すべてのデータベース・パスワードは大文字と小文字が区別されます。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、信頼できる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、信頼できる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。ユーザ ID またはパスワードを指定しない場合は、それぞれ DBA と sql が使用されます。

Ultra Light データベースのアップグレード・パス

Ultra Light ではデータベースの作成方法が複数あるので、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と、その実行方法を示します。

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)

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](#)」 392 ページを参照してください。

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については、「[Ultra Light のデータベース・アップグレード・ツール](#)」 422 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、次のいずれかの方法を選択できます。

- **データベースのアップグレード・ウィザード** 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。**データベースのアップグレード・ウィザード**の指示に従う前に、次の点を決定します。
 - 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。
 - **新しい Ultra Light 11.0.0 データベース** データベースを作成して、そのデータベースに接続する場合は、このオプションを選択します。
 - **既存の Ultra Light 11.0.0 データベース** 古いデータベースのスキーマを新しいデータベースに適用する場合は、このオプションを選択します。
- **ulunloadold ユーティリティ** 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
 - バッチ指向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 11.0.0 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. データベースのバックアップなど、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](#)」 395 ページを参照してください。
2. [スタート] - [プログラム] - [SQL Anywhere 11] - [Sybase Central] を選択します。
3. [ツール] - [Ultra Light 11] - [データベースのアップグレード] を選択します。
4. [アップグレードするデータベース] ページで、次のいずれかのオプションを選択します。
 - **[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. 接続パラメータを指定し、[次へ] をクリックします。
6. [アップグレード先] ページで、次のいずれかのオプションを選択します。
 - **[新しいデータベース]** 新しいデータベース・ファイルを作成して、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] ウィンドウを開き、既存の Ultra Light 11.0.0 データベースに接続します。

7. データベースのアップグレード・ウィザードの指示に従います。[完了] をクリックします。

以前のバージョンの Ultra Light に信頼できるルート証明書が含まれていた場合は、新しい Ultra Light データベースに追加します。

以前のバージョンの Ultra Light データベースのユーザが新しいデータベースに表示されない場合は、ユーザを追加します。「[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 11.0.0 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. データベースのバックアップなど、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](#)」 395 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 11.0.0 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](#)」 394 ページを参照してください。
3. 古いデータベースのアンロード・ユーティリティ (ulunloadold) を実行し、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
```

次のことを確認します。

- ulunloadold ユーティリティによって作成される XML ファイルの名前を指定済みである。
- 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* に DBF と schema_file パラメータのいずれかを指定済みである。

それ以外のすべてのオプションは任意です。

ulunloadold ユーティリティのサポートされているオプションの詳細については、「[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 を新しいデータベースにロードする場合は、**-c connection-string** オプションによって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) が設定されます。

指定する **-o [extended-options]** は、データベースの特性やプロパティを変更するかどうか (大文字と小文字を区別するデータベースを大文字と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 8.0.x のスキーマ・ファイルを、*db.udb* という Ultra Light バージョン 11.0.0 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
```

バージョン 9 以前の Ultra Light アプリケーション・コードをバージョン 11.0.0 に移植する

以前の Ultra Light アプリケーションは、新しいバージョン 11.0.0 の API で再構築する必要があります。Ultra Light 10 のリリース以来、これらの API は大幅に強化されました。アプリケーションを再構築する前に、必要に応じてコードの変更を行ってください。

作業を開始する前に、Ultra Light の新機能や動作の変更を調べて、使用している API への変更点を確認します。「[Ultra Light の新機能](#)」 270 ページ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に関する特別な注意事項

- 接続コードを更新してください。
 - 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.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』
- バージョン 10 で、データベースの作成に個別の *.usm* ファイルは不要になりました。代わりに、データベースをデスクトップで設定してアプリケーションを使用して展開するか、アプリケーションがプログラムでデータベースを作成できます。
- 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。
- Embedded SQL ファイルが複数ある場合は、SQL プリプロセッサ (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。
- Unicode 文字のサポートは、前のバージョンから変更されています。バージョン 11.0.0 の Ultra Light データベースでは、マルチバイト文字に UTF-8 エンコードが使用されています。その結果、非 Unicode ランタイムで実行している Unicode のデータベースを考慮する必要がなくなりました。

バージョン 9 以前の 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 や方法があるので、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と、その実行方法を示します。

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. Ultra Light データベースをアップグレードします。 2. アプリケーションのデータ・アクセス・コードを書き換えます。 注意 SQL Anywhere 10 の C++ API 移行ウィザードを使用することもできますが、それでもアプリケーションのデータ・アクセス・コードの一部を書き換える必要がある場合があります。 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。 C++ API 移行ウィザード は SQL Anywhere 10 で使用できます。SQL Anywhere 10 Developer Edition は、http://www.ianywhere.jp/dl/dl_ev1.html からダウンロードできます。
Embedded SQL アプリケーション	変更された点のごくわずかで、ツールは不要です。
Java アプリケーション	Ultra Light J を参照してください。

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 バージョン 10 以降では、スキーマがデータベースに統合されました。ULConnection オブジェクトの OpenWithCreate 関数を書き換えるために必要なすべてのコンポーネントと、すべてのスキーマ・アップグレード・コードが削除されました。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。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してください。詳細については、「初回接続時での Ultra Light データベースの作成」『Ultra Light データベース管理とリファレンス』と「Ultra Light データベースの作成」『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する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したり、開いたり、削除したり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](#)」 392 ページを参照してください。

バージョン 9 以前の SQL Anywhere データベースから Ultra Light データベースを抽出する

バージョン 10.0.0 以降の SQL Anywhere データベースから Ultra Light データベースを抽出するには、**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。
- バッチ指向の処理を実装する場合は、ユーティリティを選択します。

◆ **バージョン 9 以前の 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 (Sybase Central の場合)。**

1. データベースのバックアップなど、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](#)」 395 ページを参照してください。
2. SQL Anywhere データベースをアップグレードして、Ultra Light での使用を考慮したうえでそのデータベースを準備します。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。

バージョン 9 以前の SQL Anywhere データベースのアップグレードについては、「[バージョン 9 以前のデータベースをバージョン 11 用に再構築](#)」 399 ページを参照してください。

3. [スタート] - [プログラム] - [SQL Anywhere 11] - [Sybase Central] を選択して、Sybase Central を起動します。
4. [ツール] - [Ultra Light 11] - [(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。
5. データベース抽出ウィザードの指示に従います。

◆ バージョン 9 以前の 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. データベースのバックアップなど、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](#)」 395 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 11.0.0 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](#)」 394 ページを参照してください。
3. 既存の SQL Anywhere データベースをアップグレードして、Ultra Light での使用を考慮したうえでそのデータベースを準備します。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。

バージョン 9 以前の SQL Anywhere データベースのアップグレードについては、「[バージョン 9 以前のデータベースをバージョン 11 用に再構築](#)」 399 ページを参照してください。

4. データベースの初期化ユーティリティ (ulinit) を実行して、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 "SA-connection-string" -c "UL-connection-string"
-n pubname [ options ]
```

次のことを確認します。

- 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である。
- Ultra Light データベースに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である。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。「[Ultra Light で使用するデータベース作成パラメータの選択](#)」『[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 データベース初期化ユーティリティ \(ulinit\)](#)」
『[Ultra Light データベース管理とリファレンス](#)』を参照してください。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・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。

SQL Remote 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。

- **ソフトウェア・アップグレードは一度に 1 サイトずつ実行できる** 旧バージョンの Message Agent (dbremote) は、バージョン 11 の Message Agent とメッセージを交換することができます。バージョン 5 の SQL Remote では、compression データベース・オプションの値が -1 (マイナス 1) に設定されている場合、バージョン 5 の Message Agent とバージョン 11 の Message Agent で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。
- **データベースのアップグレード** SQL Anywhere バージョン 9 以前を使用していたリモート・データベースまたは統合データベースをアップグレードする場合は、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・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。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、「[バージョン 9 以前のデータベースをバージョン 11 用に再構築](#)」 399 ページを参照してください。
- **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 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.sybase.com/detail?id=1034174#335> を参照してください。
- **SQL Remote のアップグレード** バージョン 11.0.0 では、SQL Remote の VIM と MAPI の各メッセージ・システムがサポートされなくなりました。VIM または MAPI を使用するデータベースを SQL Anywhere バージョン 11 にアップグレードする場合は、メッセージ・タイプを File、FTP、または SMTP に変更してください。メッセージ・タイプが MAPI または VIM の場合、dbremote.exe は起動しません。

例

以下に、バージョン 5 の SQL Remote の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てから、統合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・ファイル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 の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。統合データベースのアンロードと再ロードの方法については、「[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。

2. リモート・データベース・サーバと **Message Agent** を 1 つずつアップグレードしてから、リモート・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・ファイル・フォーマット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 以外の値に設定すると、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コード化機能を利用できます。リモート・データベースのアンロードと再ロードの方法については、[『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。
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

- .cab ファイル
 - バージョン 10.0.0 の強化, 158
- .cdb
 - ファイル拡張子、バージョン 10.0.0 でサポート対象外, 194
- .NET
 - バージョン 11.0.0 での動作の変更, 100
- .NET および C++ 用の dbmlsync API
 - バージョン 11.0.0 の新機能, 75
- .sasrv.ini
 - バージョン 10.0.0 での動作の変更, 166
- .scRepository
 - バージョン 11.0.0 での名前の変更, 92
- .wrt
 - ファイル拡張子、バージョン 10.0.0 でサポート対象外, 194
- @data オプション
 - バージョン 11.0.0 の強化, 36
- @filename オプション
 - バージョン 10.0.0 の強化, 137
- \$ml_connect
 - バージョン 10.0.0 の新機能, 214
- \$ml_password
 - バージョン 10.0.0 の新機能, 214
- \$ml_user
 - バージョン 10.0.0 の新機能, 214
- #hook_dict テーブル
 - バージョン 10.0.0 での dbmlsync の強化, 211
 - バージョン 10.0.0 での SQL Remote の強化, 231
- % 演算子
 - バージョン 11.0.0 での動作の変更, 69
- 1252NOR 照合
 - バージョン 10.0.0 の新機能, 163
- 1254TRKALT 照合
 - バージョン 9.0.1 の新機能, 291
- 256 ビットの AES 暗号化
 - バージョン 11.0.0 の新機能, 34
- 508 条
準拠, 364
- 64 ビット
 - バージョン 11.0.0 での動作の変更, 66
- 64 ビットのプラットフォーム、Mobile Link でのサポート
 - バージョン 11.0.0 の新機能, 74
- 950TWN 照合
 - バージョン 9.0.2 で廃止予定, 277
- bc オプション
 - バージョン 10.0.0 で削除された Mobile Link [mlsrv10] の機能, 216
- bn オプション
 - バージョン 10.0.0 での Mobile Link [mlsrv10] の動作の変更, 217
- cc オプション
 - バージョン 9.0.1 のデータベース・サーバの新機能, 289
- ch オプション
 - 10.0.0 でのデータベース・サーバの動作の変更, 173
- cl オプション
 - 10.0.0 でのデータベース・サーバの動作の変更, 173
- cm オプション
 - バージョン 10.0.0 の Mobile Link [mlsrv10] の新機能, 204
 - バージョン 10.0.0 の新機能, 141
 - バージョン 11.0.1 の dbunload の新機能, 3
- codepage オプション
 - バージョン 11.0.0 で廃止予定, 96
- cp オプション
 - バージョン 11.0.0 での dbunload の強化, 36
- cr オプション
 - バージョン 9.0.1 のデータベース・サーバの新機能, 289
- cs オプション、Mobile Link (mlsrv11)
 - バージョン 11.0.0 の新機能, 74
- ct オプション
 - 10.0.0 でのデータベース・サーバの動作の変更, 167
- cv オプション
 - バージョン 9.0.1 のデータベース・サーバの新機能, 289
- c オプション
 - 10.0.0 でのデータベース・サーバの動作の変更, 173
- dc オプション
 - 9.0.1 の新機能 dbmlsync, 295
- dd オプション

- バージョン 10.0.0 で削除された Mobile Link [mlsrv10] の機能, 216
- dh オプション
 - バージョン 10.0.0 の新機能, 141
- dsd オプション
 - バージョン 10.0.0 での Mobile Link の新機能, 206
- ds オプション
 - バージョン 10.0.1 の新機能, 109
- dt オプション
 - バージョン 10.0.0 の新機能, 141
- d オプション
 - バージョン 10.0.0 で削除された Mobile Link [mlsrv10] の機能, 216
 - バージョン 10.0.0 で削除されたデータベース・サーバ・サポート, 194
- ec オプション
 - バージョン 10.0.0 での動作の変更, 176
- esu オプション
 - バージョン 10.0.0 での Mobile Link の新機能, 206
- es オプション
 - バージョン 11.0.0 の新機能, 37
- e オプション
 - バージョン 10.0.1 で廃止予定の初期化ユーティリティ [dbinit] のオプション, 114
 - バージョン 11.0.0 でサポート終了される初期化ユーティリティ [dbinit] のオプション, 69
- fd オプション
 - バージョン 10.0.0 の QAnywhere の新機能, 227
- fr オプション
 - バージョン 10.0.0 の QAnywhere の新機能, 227
- ftr オプション
 - バージョン 10.0.0 での Mobile Link [dbmlsrv10] の新機能, 205
- ga オプション、Mobile Link [dblsn]
 - バージョン 11.0.0 で廃止予定, 77
- gb オプション
 - バージョン 11.0.0 の強化, 38
- gi オプション、Mobile Link [dblsn]
 - バージョン 11.0.0 で廃止予定, 77
- gss オプション
 - バージョン 11.0.0 での動作の変更, 62
 - バージョン 11.0.0 の強化, 53
- gtc オプション
 - バージョン 10.0.0 の新機能, 141
- gx オプション
 - バージョン 10.0.1 で廃止予定のデータベース・サーバのオプション, 114
 - バージョン 11.0.0 でサポート終了, 62
- g オプション
 - バージョン 10.0.0 で削除された Listener [dblsn] のオプション, 213
 - バージョン 11.0.0 での dbunload の強化, 36
- g オプション、Mobile Link [dblsn]
 - バージョン 11.0.0 で廃止予定, 77
- idl オプション
 - バージョン 10.0.1 の QAnywhere の新機能, 118
- id オプション、QAnywhere (qastop)
 - バージョン 11.0.0 の新機能, 78
- im オプション
 - バージョン 11.0.0 の新機能, 38
- ja オプション
 - バージョン 10.0.0 の dbinit で廃止予定, 170
 - バージョン 10.0.0 の dbupgrad で廃止予定, 170
- jconnect オプション
 - バージョン 10.0.0 で dbconsole に使用できない, 252
 - バージョン 10.0.0 で Interactive SQL に使用できない, 252
- jdk オプション
 - バージョン 10.0.0 の dbinit で廃止予定, 170
 - バージョン 10.0.0 の dbupgrad で廃止予定, 170
- jr オプション
 - バージョン 10.0.0 の dbunload で廃止予定, 170
 - バージョン 10.0.0 の dbupgrad で廃止予定, 170
- j オプション
 - バージョン 10.0.0 の dbupgrad で廃止予定, 170
- ksc オプション
 - バージョン 11.0.0 の新機能, 38
- ksd オプション
 - バージョン 11.0.0 の新機能, 38
- ks オプション
 - バージョン 11.0.0 の新機能, 38
- k オプション
 - バージョン 10.0.0 で廃止予定の Mobile Link [dbmlsync] のオプション, 223
- la_port オプション
 - バージョン 10.0.0 で廃止予定の QAnywhere オプション, 229
- lp オプション
 - バージョン 10.0.0 の QAnywhere の新機能, 229
- lsc オプション、Mobile Link (mlsrv11)
 - バージョン 11.0.0 の新機能, 74

-
- l オプション
バージョン 11.0.1 の dbunload の新機能, 3
 - mn オプション
バージョン 10.0.0 の QAnywhere [qaagent] の新機能, 228
 - mp オプション
バージョン 10.0.0 の QAnywhere [qaagent] の新機能, 228
 - mu オプション
バージョン 10.0.0 の QAnywhere [qaagent] の新機能, 228
 - nc オプション
バージョン 10.0.0 での Mobile Link [dbmlsrv10] の新機能, 205
 - ni オプション
dblsn、バージョン 11.0.0 の新機能, 76
バージョン 10.0.0 での Listener [dblsn] の新機能, 213
 - no オプション
バージョン 11.0.0 での dbunload の強化, 36
 - ns オプション
バージョン 10.0.0 での Listener [dblsn] の新機能, 213
 - nu オプション
dblsn、バージョン 11.0.0 の新機能, 76
バージョン 10.0.0 での Listener [dblsn] の新機能, 213
 - odbc オプション
バージョン 10.0.0 で dbconsole に使用できない, 252
バージョン 10.0.0 で Interactive SQL に使用できない, 252
 - os オプション
バージョン 10.0.0 での動作の変更, 176
 - ot オプション
バージョン 10.0.0 の新機能, 141
 - oy オプション
バージョン 10.0.0 で削除された Mobile Link [mlsrv10] の機能, 218
 - pc オプション
バージョン 10.0.0 での Listener [dblsn] の新機能, 213
バージョン 10.0.0 での Mobile Link [dbmlsync] の新機能, 211
バージョン 10.0.0 の QAnywhere の新機能, 227
 - pc- オプション
dblsn、バージョン 11.0.0 の新機能, 76
 - policy オプション
バージョン 10.0.0 でのデフォルトの変更, 229
 - port オプション
バージョン 10.0.0 で削除された QAnywhere オプション, 229
 - ppv オプション
バージョン 11.0.1 の Mobile Link (mlsrv11) の強化, 14
 - push_notifications オプション (参照 -push オプション)
バージョン 10.0.0 で名前が変更された QAnywhere オプション, 229
 - push オプション
バージョン 10.0.0 の QAnywhere の新機能, 227
 - p オプション
バージョン 10.0.0 で削除された dblic サポート, 194
 - qc オプション
バージョン 10.0.0 での Mobile Link [dbmlsync] の新機能, 223
 - qn オプション
バージョン 10.0.0 の新機能, 163
 - r オプション
バージョン 10.0.0 での Listener [dblsn] の新機能, 213
 - sf オプション
バージョン 10.0.0 の新機能, 136
バージョン 11.0.0 の強化, 38, 52
 - sk オプション
バージョン 10.0.0 の新機能, 136
 - sm オプション
バージョン 10.0.0 での Mobile Link [dbmlsrv10] の新機能, 205
バージョン 11.0.0 の新機能, 32
バージョン 11.0.1 での Mobile Link の動作の変更, 16
 - sn オプション
バージョン 10.0.0 の新機能, 129
 - sp オプション
dbmlsync、バージョン 11.0.0 の新機能, 75
 - sur オプション
バージョン 10.0.0 の QAnywhere の新機能, 227
 - su オプション
バージョン 10.0.0 の新機能, 141
 - sv オプション
バージョン 11.0.1 の Mobile Link (dblsn) の強化, 14

- s オプション
 - バージョン 9.0.1 の新しいバックアップ・ユーティリティ・オプション, 290
- tc オプション、Mobile Link (mlsrv11)
 - バージョン 11.0.0 の新機能, 74
- tf オプション、Mobile Link (mlsrv11)
 - バージョン 11.0.0 の新機能, 75
- tu オプション
 - 9.0.1 の新機能 dbmlsync, 295
- ua オプション
 - バージョン 9.0.1 の新機能, 293
- uc サーバ・オプション
 - バージョン 11.0.0 での動作の変更, 68
- uf オプション
 - バージョン 10.0.0 の新機能, 159
- ui サーバ・オプション
 - バージョン 11.0.0 での動作の変更, 68
- um オプション
 - バージョン 11.0.0 の新機能, 38
- dt オプション
 - バージョン 10.0.0 での Mobile Link の新機能, 206
- us オプション
 - 9.0.1 の新機能 dbmlsync, 295
 - バージョン 10.0.0 で削除された Mobile Link [mlsrv10] の機能, 215
- ux オプション
 - 10.0.0 での Mobile Link [dbmlsync] の新機能, 208
 - 10.0.0 での Mobile Link [mlsrv10] の新機能, 208
 - 10.0.0 での SQL Remote [dbremote] の新機能, 231
- u オプション
 - バージョン 10.0.0 で削除された Mobile Link [mlsrv10] の機能, 216
- version
 - バージョン 11.0.0 での dbisql の新機能, 91
- ve オプション
 - バージョン 10.0.0 での Mobile Link [dbmlsrv10] の新機能, 205
- vi オプション
 - バージョン 11.0.1 の Mobile Link (mlsrv11) の強化, 14
- vm オプション
 - バージョン 11.0.1 の Mobile Link (mlsrv11) の強化, 14
- vq オプション
 - バージョン 11.0.1 の Mobile Link (mlsrv11) の強化, 14
- vr オプション
 - バージョン 10.0.0 での Mobile Link [mlsrv10] の動作の変更, 217
- vt オプション
 - バージョン 10.0.0 での Mobile Link [mlsrv10] の動作の変更, 217
- vu オプション
 - バージョン 10.0.0 での Mobile Link [mlsrv10] の動作の変更, 217
- wu オプション
 - バージョン 10.0.0 での Mobile Link [mlsrv10] の動作の変更, 216
- w オプション
 - バージョン 10.0.0 での Mobile Link [mlsrv10] の動作の変更, 216
- xd オプション
 - バージョン 10.0.1 の QAnywhere の新機能, 118
 - バージョン 11.0.1 のデータベース・サーバの新機能, 7
- xf オプション
 - バージョン 10.0.0 の新機能, 129
- xo オプション
 - バージョン 10.0.0 の Mobile Link [mlsrv10] の新機能, 204
- xp オプション
 - バージョン 10.0.0 の新機能, 130
 - バージョン 11.0.0 での動作の変更, 64
- xs オプション
 - バージョン 10.0.0 での動作の変更, 176
- x オプション
 - バージョン 10.0.0 の Mobile Link [mlsrv10] の新しい構文, 204
- y オプション
 - バージョン 10.0.0 で削除されたデータベース・サーバ・オプション, 194
 - バージョン 10.0.0 で削除されたデータベース・サーバ・サポート, 194
- zac オプション
 - バージョン 10.0.0 で削除された Mobile Link [mlsrv10] の機能, 218
- za オプション
 - バージョン 10.0.0 で削除された Mobile Link [mlsrv10] の機能, 218
- zd オプション

バージョン 9 で削除された Mobile Link
[dbmlsrv] の機能, 329

-zec オプション
バージョン 10.0.0 で削除された Mobile Link
[mlsrv10] の機能, 218

-zl オプション
バージョン 10.0.1 での動作の変更, 111

-zoc オプション
バージョン 11.0.0 の新機能, 52

-zp オプション
バージョン 10.0.0 の新機能, 141

-zus オプション
バージョン 10.0.0 の Mobile Link [mlsrv10] の新
機能, 205

A

a_backup_db 構造体
backup_writefile メンバ、バージョン 10.0.0 でサ
ポート対象外, 195

a_compress_db 構造体
バージョン 10.0.0 でサポート対象外, 195

a_db_collation 構造体
バージョン 10.0.0 でサポート対象外, 193

a_db_info 構造体
compressed メンバ、バージョン 10.0.0 でサポー
ト対象外, 195
wrtbufsize メンバ、バージョン 10.0.0 でサポー
ト対象外, 195
wrtnamebuffer メンバ、バージョン 10.0.0 でサ
ポート対象外, 195

a_dblic_info 構造体
バージョン 10.0.1 での動作の変更, 124

a_stats_line 構造体
バージョン 10.0.0 でサポート対象外, 195

a_validate_type 列挙体
バージョン 10.0.0 で廃止予定の
VALIDATE_DATA パラメータ, 174
バージョン 10.0.0 で廃止予定の
VALIDATE_FULL パラメータ, 174
バージョン 10.0.0 で廃止予定の
VALIDATE_INDEX パラメータ, 174
バージョン 11.0.0 の強化, 56

a_writefile 構造体
バージョン 10.0.0 でサポート対象外, 195

AccentSensitive プロパティ
バージョン 10.0.0 の新機能, 146

ACCENT 句

CREATE DATABASE 文、バージョン 10.0.1 で
廃止予定, 114

ActiveSync
バージョン 10.0.0 での Mobile Link の動作の変
更, 223

ActiveSync プロバイダのアクティビティ・ログ
9.0.1 の新機能, 295

Adaptive Server Anywhere (参照 SQL Anywhere)
バージョン 10.0.0 での SQL Anywhere への名前
の変更, 165
バージョン 11 へのアップグレード, 391

Adaptive Server Enterprise
バージョン 10.0.0 で SQL Remote サポートの削
除, 231
バージョン 10.0.0 での ODBC ドライバの動作
の変更, 258

Adaptive Server Enterprise の互換性
10.0.0 での動作の変更, 196

addBatch メソッド
バージョン 10.0.0 の新機能, 156

ADO.NET 2.0 のサポート
バージョン 10.0.0 の新機能, 156

adsodbc サーバ・クラス
バージョン 11.0.0 の新機能, 55

AES256_FIPS 暗号化アルゴリズム
バージョン 11.0.0 の新機能, 34

AES256 暗号化アルゴリズム
バージョン 11.0.0 の新機能, 34

AGENT 接続パラメータ
バージョン 8.0.1 の廃止予定機能, 356

allow_read_client_file オプション
バージョン 11.0.0 の新機能, 37

allow_read_client_file プロパティ
バージョン 11.0.0 の新機能, 39

allow_snapshot_isolation オプション
バージョン 10.0.0 の新機能, 130

allow_snapshot_isolation プロパティ
バージョン 10.0.0 の新機能, 143

allow_write_client_file オプション
バージョン 11.0.0 の新機能, 37

allow_write_client_file プロパティ
バージョン 11.0.0 の新機能, 39

ALTER DATABASE 文
バージョン 10.0.0 での動作の変更, 170, 191
バージョン 10.0.1 の強化, 107
バージョン 11.0.0 の強化, 44
バージョン 11.0.1 の強化, 4

- ALTER DBSPACE 文
バージョン 10.0.1 での動作の変更, 110
- ALTER EVENT 文
バージョン 10.0.0 での動作の変更, 191
バージョン 11.0.0 の強化, 46, 47
- ALTER FUNCTION 文
バージョン 11.0.0 の強化, 44
- ALTER INDEX 文
バージョン 10.0.0 の強化, 154
- ALTER INDEX 文の REBUILD 句
バージョン 10.0.0 の強化, 152
- ALTER LOGIN POLICY 文
バージョン 11.0.0 の新機能, 45
- ALTER MATERIALIZED VIEW 文
バージョン 10.0.0 の新機能, 152
バージョン 11.0.0 の強化, 44
- AlternateMirrorServerName プロパティ
バージョン 11.0.0 の新機能, 41
- AlternateServerName プロパティ
バージョン 10.0.0 の新機能, 146
- ALTER PROCEDURE 文
バージョン 11.0.0 の強化, 44
- ALTER PUBLICATION 文
バージョン 10.0.0 での Ultra Light の強化, 238
- ALTER PUBLICATION 文 [Mobile Link] [SQL Remote]
バージョン 10.0.0 での動作の変更, 191
- ALTER SERVER 文
バージョン 10.0.0 での動作の変更, 191
- ALTER SERVICE 文
バージョン 10.0.0 の強化, 161
- ALTER STATISTICS 文
バージョン 10.0.0 の新機能, 152
- ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]
バージョン 10.0.0 での動作の変更, 191
- ALTER SYNCHRONIZATION USER 文 [Mobile Link]
バージョン 10.0.0 での動作の変更, 191
- ALTER TABLE 文
バージョン 10.0.0 での動作の変更, 191
バージョン 10.0.0 の強化, 152
バージョン 9.0.1 の強化, 288
- ALTER TEXT CONFIGURATION 文
バージョン 11.0.0 の新機能, 46
- ALTER TEXT INDEX 文
バージョン 11.0.0 の新機能, 46
- ALTER USER 文
バージョン 11.0.0 の新機能, 45
- ALTER VIEW 文
バージョン 9.0.1 の強化, 288
- ALTER WRITEFILE 文
バージョン 10.0.0 でサポート対象外, 195
- an_expand_db 構造体
バージョン 10.0.0 でサポート対象外, 195
- ansi_blanks オプション
バージョン 10.0.0 での動作の変更, 174
- ansi_integer_overflow オプション
バージョン 10.0.0 での動作の変更, 175
バージョン 11.0.0 でサポート終了, 69
- ansi_nulls オプション
バージョン 9.0.1 の強化, 288
- ansi_substring オプション
11.0.1 で非推奨の取り消し, 13
バージョン 10.0.0 の新機能, 140
バージョン 11.0.0 でサポート終了, 69
- ansi_substring プロパティ
バージョン 10.0.0 の新機能, 143
- AppInfo 接続パラメータ
バージョン 10.0.0 の強化, 134
バージョン 10.0.1 の強化, 112
バージョン 11.0.0 の強化, 33
- ApproximateCPUTime プロパティ
バージョン 10.0.0 の新機能, 143
- APP 接続パラメータ
バージョン 11.0.0 の強化, 33
- ArbiterState プロパティ
バージョン 10.0.0 の新機能, 146
- asademo.db ファイル
バージョン 10.0.0 での名前の変更, 257
- ASAJDBC
バージョン 10.0.0 の名前の変更, 191
- ASANYSH 環境変数
バージョン 10.0.0 での名前の変更, 199
- ASANY 環境変数
バージョン 10.0.0 での名前の変更, 199
- ASAODBC
バージョン 10.0.0 の名前の変更, 191
- ASAProv
バージョン 10.0.0 での動作の変更, 169
- ASCII
バージョン 10.0.0 での Ultra Light の強化, 241
- ASP.NET プロバイダ
バージョン 11.0.1 の新機能, 5

`assume_distinct_servers` オプション
バージョン 9.0.2 で廃止予定, 277

ATTACH TRACING 文
バージョン 10.0.0 の新機能, 152

AuditingTypes プロパティ
バージョン 10.0.0 の新機能, 146

`authenticate_parameters`
バージョン 10.0.0 での動作の変更, 215

`authenticate_user`
バージョン 10.0.0 での動作の変更, 215

`authenticate_user_hashed`
バージョン 10.0.0 での動作の変更, 215

`authenticate.sql`
バージョン 10.0.1 の新機能, 111

Authenticated プロパティ
接続プロパティ、11.0.1 の新機能, 2
データベース・プロパティ、11.0.1 の新機能, 2

AuthType プロパティ
バージョン 11.0.0 の新機能, 39

AUTOINCREMENT
`reset.sql` スクリプトを使用して次に使用可能な値を保持, 397
再構築されたデータベースで次に使用可能な値を保持, 397

`automatic_timestamp` オプション
バージョン 11.0.0 でサポート終了, 69

AWE キャッシュ
バージョン 10.0.0 の強化, 141

B

`background_priority` オプション
バージョン 11.0.0 で廃止予定, 71

BACKUP 権限
バージョン 10.0.0 の新機能, 136

BACKUP 文
バージョン 10.0.0 での動作の変更, 192
バージョン 10.0.0 の強化, 135
バージョン 11.0.1 の強化, 7
バージョン 9.0.1 の強化, 290
バージョン 9.0.1 の変数の強化サポート, 287

BASE64_DECODE 関数
バージョン 9.0.1 の新機能, 287

BASE64_ENCODE 関数
バージョン 9.0.1 の新機能, 287

`begin_connection`
バージョン 10.0.0 での動作の変更, 215

BEGIN SNAPSHOT 文
バージョン 11.0.0 の新機能, 46

`begin` スクリプト
バージョン 10.0.0 での Mobile Link の動作の変更, 215

BIGINT データ型
バージョン 10.0.0 での動作の変更, 169

bin32 ディレクトリ
バージョン 11.0.0 での動作の変更, 100

bin64 ディレクトリ
バージョン 11.0.0 での動作の変更, 100

BINSEARCH プロトコル・オプション
バージョン 11.0.0 でサポート終了, 69

BIT_AND 関数
バージョン 10.0.0 の新機能, 149

BIT_LENGTH 関数
バージョン 10.0.0 の新機能, 149

BIT_OR 関数
バージョン 10.0.0 の新機能, 149

BIT_SUBSTR 関数
バージョン 10.0.0 の新機能, 149

BIT_XOR 関数
バージョン 10.0.0 の新機能, 149

BlackBerry
`ULjDbT` ユーティリティ, 20
`Ultra Light J` データベース転送ユーティリティ, 20
`Ultra Light` バージョン 11.0.0 での強化, 86

BLOB
問い合わせ、バージョン 11.0.0 の新機能, 32
バージョン 10.0.0 での `Ultra Light` の強化, 235
バージョン 10.0.0 の強化, 133

BlobArenas プロパティ
バージョン 10.0.0 で廃止予定のデータベース・プロパティ, 199

BOM (バイト順マーク)
バージョン 11.0.0 の強化, 95

BREAK 文
バージョン 10.0.0 の新機能, 154

`buffer_size` オプション
10.0.0 での Mobile Link クライアントの新しいプロトコル・オプション, 209

C

C++ API 移行ウィザード
`Ultra Light` での使用, 425
バージョン 11.0.0 でサポート終了, 87

- C++ インタフェースの混在
 - 9.0.1 の Ultra Light の新機能, 297
- C2 プロパティ
 - バージョン 10.0.0 で削除, 194
- CacheHitsEng プロパティ
 - バージョン 11.0.0 での名前の変更, 67
- CacheHits プロパティ
 - バージョン 11.0.0 の新機能, 67
- CachePinned プロパティ
 - バージョン 10.0.0 の新機能, 145
- CacheReadEng プロパティ
 - バージョン 10.0.0 の新機能, 145
 - バージョン 11.0.0 での名前の変更, 67
- CacheReadWorkTable プロパティ
 - バージョン 11.0.0 の新機能, 39, 41
- CacheRead プロパティ
 - バージョン 11.0.0 の新機能, 67
- CacheSizingStatistics プロパティ
 - バージョン 10.0.0 の新機能, 150
- CALIBRATE PARALLEL READ 句
 - バージョン 10.0.0 の強化, 152
- CarverHeapPages プロパティ
 - バージョン 10.0.0 の新機能, 145
- CASE 句
 - CREATE DATABASE 文、バージョン 10.0.1 で廃止予定, 114
- CASE 式
 - Ultra Light バージョン 11.0.0 での強化, 85
 - バージョン 11.0.0 の強化, 46
- CASE 文
 - Ultra Light バージョン 11.0.0 での強化, 85
 - バージョン 11.0.0 の強化, 46
- CAST 関数
 - バージョン 10.0.0 での動作の変更, 175
- CatalogCollation プロパティ
 - バージョン 10.0.1 の新機能, 106
- CATALOGS ローセット、OLE DB
 - バージョン 11.0.1 の新機能, 6
- Certicom
 - 2002 年 7 月で証明書の発行終了, 345
 - Security Builder GSE のバージョン、バージョン 10.0.1 の強化, 124
- certificate_password プロトコル・オプション
 - バージョン 11.0.0 でサポート終了, 100
 - バージョン 11.0.0 での動作の変更, 76
- certificate プロトコル・オプション
 - バージョン 11.0.0 でサポート終了, 100
 - バージョン 11.0.0 での動作の変更, 76
- バージョン 11.0.0 での動作の変更, 76
- CharSet プロパティ
 - バージョン 10.0.0 の強化, 131
- CHECKPOINT 文
 - バージョン 10.0.1 での Ultra Light の強化, 122
- Checksum プロパティ
 - バージョン 9.0.1 の新機能, 288
- CleanablePagesAdded プロパティ
 - バージョン 10.0.0 の新機能, 146
- CleanablePagesCleaned プロパティ
 - バージョン 10.0.0 の新機能, 146
- ClientNodeAddress プロパティ
 - バージョン 11.0.0 の新機能, 39
- ClientStmtCacheHits プロパティ
 - バージョン 10.0.1 の新機能, 104
- ClientStmtCacheMisses プロパティ
 - バージョン 10.0.1 の新機能, 104
- ClusteredIndexes プロパティ
 - バージョン 10.0.0 で廃止予定のデータベース・プロパティ, 199
- CodeWarrior
 - サポートされるバージョン, 331
- CodeWarrior ステーションナリ
 - 9.0.1 の Ultra Light C++ コンポーネントの新機能, 298
- collect_statistics_on_dml_updates オプション
 - バージョン 10.0.0 の新機能, 140
- collect_statistics_on_dml_updates プロパティ
 - バージョン 10.0.0 の新機能, 143
- CollectStatistics プロパティ
 - バージョン 10.0.0 の新機能, 150
- CommBufferSize 接続パラメータ
 - バージョン 10.0.0 の強化, 134
- COMMENT 文
 - 廃止予定の句, 13
 - バージョン 10.0.0 での動作の変更, 192
 - バージョン 10.0.0 の強化, 153
 - バージョン 11.0.0 の強化, 46, 47
- COMMIT 文
 - 9.0.1 の Ultra Light 動的 SQL の新機能, 297
- Common Access Card
 - バージョン 11.0.1 の新機能, 14
- Common Access Card を使用したクライアント認証
 - バージョン 11.0.1 の新機能, 14
- COMPARE 関数
 - バージョン 10.0.1 の強化, 106
- CompressedBTrees プロパティ

バージョン 10.0.0 で廃止予定のデータベース・プロパティ, 199

CompressionThreshold 接続パラメータ
バージョン 10.0.0 の強化, 134

Compression プロパティ
バージョン 10.0.0 でサポート対象外, 196

COMPRESS 関数
バージョン 10.0.0 の強化, 151
バージョン 9.0.1 の新機能, 287

confirmation_handler
バージョン 10.0.0 の新機能, 213

conflicted_deletes
バージョン 10.0.0 の新しい動作, 218

conflicted_inserts
バージョン 10.0.0 の新しい動作, 218

conflicted_updates
バージョン 10.0.0 の新しい動作, 218

conn_auditing オプション
バージョン 10.0.0 の新機能, 134

conn_auditing プロパティ
バージョン 10.0.0 の新機能, 143

CONNECTION_EXTENDED_PROPERTY 関数
バージョン 10.0.0 の新機能, 150

CONNECTION_PROPERTY 関数
バージョン 10.0.0 の強化, 149

ConnsDisabled プロパティ
バージョン 10.0.0 での動作の変更, 166

ConsoleLogFile プロパティ
バージョン 10.0.0 の新機能, 150
バージョン 9.0.1 の新機能, 293

ConsoleLogMaxSize プロパティ
バージョン 10.0.0 の新機能, 150

CONTAINS 探索条件
バージョン 11.0.0 の新機能, 45

contd_timeout オプション
バージョン 10.0.0 で置き換えられた Mobile Link クライアント・プロトコル・オプション, 209

CONTINUE 文
バージョン 10.0.0 の強化, 154

CORR 関数
バージョン 9.0.1 の新機能, 287

COUNT_SET_BITS 関数
バージョン 10.0.0 の新機能, 149

COVAR_POP 関数
バージョン 9.0.1 の新機能, 287

COVAR_SAMP 関数
バージョン 9.0.1 の新機能, 287

createcert ユーティリティ
バージョン 10.0.1 の新機能, 123

CREATE COMPRESSED DATABASE 文
バージョン 10.0.0 でサポート対象外, 195

CREATE DATABASE 文
バージョン 10.0.0 での BLANK PADDING 句の動作の変更, 166
バージョン 10.0.0 での動作の変更, 170
バージョン 10.0.0 の強化, 137, 153
バージョン 10.0.1 での動作の変更, 110
バージョン 10.0.1 の強化, 103, 105
バージョン 9.0.1 の強化, 288

CREATE DBSPACE 文
バージョン 10.0.1 での動作の変更, 110

CREATE DECRYPTED DATABASE 文
バージョン 11.0.1 の新機能, 4

CREATE ENCRYPTED DATABASE 文
バージョン 11.0.1 の新機能, 4

CREATE ENCRYPTED FILE 文
バージョン 10.0.0 の強化, 153

CREATE EVENT 文
バージョン 11.0.0 の強化, 47

CREATE EXPANDED DATABASE 文
バージョン 10.0.0 でサポート対象外, 195

CREATE FUNCTION 文
SET 句、バージョン 10.0.1 の新機能, 106
バージョン 11.0.1 の強化, 5

CREATE INDEX 文
9.0.1 の Ultra Light 動的 SQL の新機能, 297
バージョン 10.0.0 での動作の変更, 193

createkey ユーティリティ
バージョン 11.0.0 の新機能, 76

CREATE LOCAL TEMPORARY TABLE 文
バージョン 10.0.0 の強化, 155

CREATE LOGIN POLICY 文
バージョン 11.0.0 の新機能, 45

CREATE MATERIALIZED VIEW 文
バージョン 10.0.0 の新機能, 152
バージョン 11.0.0 の強化, 44, 46

CREATE PROCEDURE 文
SET 句、バージョン 10.0.1 の新機能, 106
バージョン 11.0.1 の強化, 5

CREATE PUBLICATION 文
バージョン 10.0.0 での Ultra Light の強化, 238

CREATE SERVER 文
バージョン 10.0.0 での動作の変更, 191

CREATE SERVICE 文

バージョン 10.0.0 の強化, 161

CREATE SYNCHRONIZATION DEFINITION 文

バージョン 10.0.0 での削除, 222

CREATE SYNCHRONIZATION SITE 文

バージョン 10.0.0 での削除, 222

CREATE SYNCHRONIZATION TEMPLATE 文

バージョン 10.0.0 での削除, 222

CREATE TABLE 文

9.0.1 の Ultra Light 動的 SQL の新機能, 297

バージョン 10.0.0 の強化, 152

バージョン 11.0.1 の強化, 5

CREATE TEXT CONFIGURATION 文

バージョン 11.0.0 の新機能, 45

CREATE TEXT INDEX 文

バージョン 11.0.0 の新機能, 46

CREATE TRIGGER 文

バージョン 11 へのアップグレードのトラブル

シューティング, 404

バージョン 11.0.1 の強化, 5

CREATE USER 文

バージョン 11.0.0 の新機能, 45

CREATE VIEW 文

バージョン 11.0.1 の強化, 5

CREATE WRITEFILE 文

バージョン 10.0.0 でサポート対象外, 195

CROSS APPLY 句

バージョン 11.0.0 の新機能, 47

CUBE 演算

バージョン 9.0.1 の新機能, 286

CUME_DIST 関数

バージョン 9.0.1 の新機能, 287

CURRENT_TIMESTAMP

バージョン 9.0.1 での動作の変更, 301

CURRENT_USER

バージョン 9.0.1 での動作の変更, 301

CurrentLineNumber プロパティ

バージョン 10.0.0 の新機能, 143

CurrentProcedure プロパティ

バージョン 10.0.0 の新機能, 143

CustDB

Ultra Light バージョン 11.0.0 での動作の変更, 87

CustDB Ultra Light サンプル

11.0.0 での動作の変更, 87

D

DataWindow .NET

バージョン 10.0.0 の新機能, 255

バージョン 10.0.1 での Vista のサポート, 126

date_format オプション

バージョン 10.0.0 での動作の変更, 175

DATEADD 関数

バージョン 10.0.0 の QAnywhere の新機能, 228

DATEPART 関数

バージョン 10.0.0 の QAnywhere の新機能, 228

DATETIME 関数

バージョン 10.0.0 の QAnywhere の新機能, 228

DB_BACKUP_WRITEFILE パラメータ

バージョン 10.0.0 でサポート対象外, 196

db_backup 関数

DB_BACKUP_WRITEFILE パラメータ、バージョン 10.0.0 でサポート対象外, 196

バージョン 10.0.0 の強化, 135

DB_CALLBACK_FINISH コールバック・パラメータ

バージョン 10.0.0 での動作の変更, 167

DB_CALLBACK_START コールバック・パラメータ

バージョン 10.0.0 での動作の変更, 167

DB_EXTENDED_PROPERTY 関数

バージョン 10.0.0 の強化, 150

バージョン 10.0.1 の強化, 106

db_locate_servers_ex 関数

バージョン 10.0.0 の強化, 157

DB_PROPERTY 関数

バージョン 10.0.0 の強化, 149

db_register_a_callback 関数

バージョン 10.0.0 での動作の変更, 167

dbasdesk.dll

バージョン 10.0.0 での mlasdesk.dll への名前の変更, 224

dbasdev.dll

バージョン 10.0.0 での mlasdev.dll への名前の変更, 224

dBase フォーマット、INPUT 文

バージョン 11.0.0 で削除されたサポート, 95

dBase フォーマット、OUTPUT 文

バージョン 11.0.0 で削除されたサポート, 95

dbasinst ユーティリティ

バージョン 10.0.0 での mlasinst への名前の変更, 223

dbbackup ユーティリティ
バージョン 10.0.0 の強化, 135
バージョン 9.0.1 の強化, 290

dbcapi.dll
11.0.0 の新機能, 48

DBChangeWriteFile 関数
バージョン 10.0.0 でサポート対象外, 195

DBCcollate 関数
バージョン 10.0.0 でサポート対象外, 193

dbcollat ユーティリティ
バージョン 10.0.0 でサポート対象外, 193

DBCompress 関数
バージョン 10.0.0 でサポート対象外, 195

dbconsole ユーティリティ
バージョン 10.0.0 の強化, 250
バージョン 11.0.0 の強化, 96
バージョン 9.0.1 の強化, 291

DBCcreatedVersion 関数
バージョン 10.0.1 の新機能, 108

DBCreateWriteFile 関数
バージョン 10.0.0 でサポート対象外, 195

DBD::ASAny
バージョン 9.0.1 の新しい Perl ドライバ, 290

DBD::ASAny
バージョン 10.0.0 での名前の変更, 157

dbdata10.dll
バージョン 10.0.1 での動作の変更, 111

dbdsn ユーティリティ
バージョン 10.0.0 での動作の変更, 193
バージョン 10.0.0 の強化, 137
バージョン 10.0.1 の強化, 109

dbelevat10.exe
バージョン 10.0.1 での Vista のサポート, 125
バージョン 10.0.1 の新機能, 125

DBExpand 関数
バージョン 10.0.0 でサポート対象外, 195

dbexpand ユーティリティ
バージョン 10.0.0 でサポート対象外, 195

dbhist ユーティリティ
バージョン 10.0.0 の強化, 137

dbinfo ユーティリティ
バージョン 10.0.0 の強化, 137

dbinit ユーティリティ
バージョン 10.0.0 での -b オプションの動作の変更, 166
バージョン 10.0.0 での動作の変更, 170
バージョン 10.0.0 の強化, 137

バージョン 10.0.1 の強化, 109
バージョン 9.0.1 の強化, 288

dbisql.exe
バージョン 11.0.0 での動作の変更, 96

dbisqlg
バージョン 11.0.0 での名前の変更, 96

dbisql ユーティリティ
バージョン 10.0.0 での動作の変更, 251
バージョン 10.0.0 の強化, 250
バージョン 11.0.0 での動作の変更, 68
バージョン 11.0.1 での動作の変更, 12

dblang ユーティリティ
バージョン 10.0.1 での動作の変更, 111

DBLIB
11.0.0 の新機能, 62

dblibtb.dll
11.0.0 の新機能, 62

dblibtw.dll
11.0.0 の新機能, 62

dblic ユーティリティ
バージョン 10.0.0 での動作の変更, 194
バージョン 10.0.1 での動作の変更, 124

dblocate ユーティリティ
バージョン 10.0.0 での動作の変更, 171
バージョン 10.0.0 の新機能, 138
バージョン 9.0.1 の強化, 290

DBLTM サービス・タイプ
バージョン 10.0.0 の新機能, 138

dbltm ユーティリティ
バージョン 10.0.0 の強化, 138

dbmlctr9.dll
バージョン 10.0.0 で Mobile Link の Windows パフォーマンス・モニタのサポートとともに削除, 224

dbmlmon ユーティリティ
バージョン 10.0.0 での mlmon への変更, 223

dbmlsrv.mle
バージョン 10.0.0 での mlsrv10.mle への変更, 224

dbmlsrv9
バージョン 10.0.0 での mlsrv10 への変更, 223

dbmlstop ユーティリティ
バージョン 10.0.0 での mlstop への変更, 223

dbmlsync -sp オプション

- バージョン 11.0.0 の新機能, 75
- dbmlsync StreamCompression 拡張オプション
 - バージョン 11.0.0 でサポート終了, 77
- dbmlsync 統合コンポーネント
 - 9.0.1 の新機能, 295
 - バージョン 11.0.0 で廃止予定, 77
- dbmlsync メッセージ・ログ・スキャン
 - バージョン 11.0.0 の強化, 75
- dbmluser ユーティリティ
 - バージョン 10.0.0 での mluser への名前の変更, 223
- dbmodenv
 - バージョン 11.0.1 での動作の変更, 12
- dbns10 ユーティリティ
 - バージョン 10.0.0 の新機能, 139
 - バージョン 11.0.0 での動作の変更, 71
- dbping ユーティリティ
 - バージョン 10.0.0 での動作の変更, 170
 - バージョン 10.0.0 の強化, 138
- dbrunsql ユーティリティ
 - バージョン 10.0.0 の新機能, 158
- dbshrink ユーティリティ
 - バージョン 10.0.0 でサポート対象外, 195
- dbspace
 - バージョン 10.0.1 の強化, 109
- dbsrv10.lic
 - バージョン 10.0.1 の新機能, 125
- DBStatusWriteFile 関数
 - バージョン 10.0.0 でサポート対象外, 195
- dbsupport ユーティリティ
 - バージョン 10.0.0 の新機能, 139, 255
 - バージョン 10.0.1 の強化, 109
 - バージョン 11.0.0 の強化, 99
- dbsvc ユーティリティ
 - バージョン 10.0.0 での動作の変更, 171
 - バージョン 10.0.0 の強化, 138, 159
 - バージョン 11.0.1 の強化, 3
- dbtlstb.dll
 - 11.0.0 の新機能, 62
- dbtlstw.dll
 - 11.0.0 の新機能, 62
- DBTools インタフェース
 - libdbtool9.so、バージョン 9.0.2 で廃止予定, 277
- dbtran ユーティリティ
 - バージョン 11.0.0 の強化, 37
- dbuleng9
 - 9.0.1 の Ultra Light の新機能, 298
- dbunload ユーティリティ
 - バージョン 10.0.0 での Ultra Light の強化, 237
 - バージョン 10.0.0 での動作の変更, 170, 171
 - バージョン 10.0.0 の強化, 139
 - バージョン 10.0.1 での動作の変更, 113
 - バージョン 11 へのアップグレードのトラブルシューティング, 404
 - バージョン 11.0.0 での動作の変更, 66
 - バージョン 11.0.0 の強化, 36
 - バージョン 11.0.1 の強化, 3
 - バージョン 9.0.1 の強化, 290
- dbupgrad ユーティリティ
 - バージョン 10.0.0 での動作の変更, 165, 170
 - バージョン 11.0.1 の強化, 3
- dbvalid ユーティリティ
 - バージョン 10.0.0 で廃止予定のオプション, 192
 - バージョン 10.0.0 の強化, 139
 - バージョン 11.0.0 の強化, 36
 - バージョン 9.0.1 での動作の変更, 302
 - バージョン 9.0.1 の強化, 288
- dbwrite ユーティリティ
 - バージョン 10.0.0 でサポート対象外, 195
- dbxtract ユーティリティ
 - バージョン 10.0.0 での変更, 232
 - バージョン 11.0.0 での動作の変更, 67
 - バージョン 11.0.0 の強化, 80
- DB 領域
 - バージョン 10.0.1 での動作の変更, 110
 - バージョン 11.0.0 での動作の変更, 63, 67
- DCX
 - 説明, viii
 - バージョン 11.0.1 の新機能, 25
- Deadlock システム・イベント
 - バージョン 11.0.0 の新機能, 56
- debug_messages オプション
 - バージョン 9.0.1 の強化, 288
- DebuggingInformation プロパティ
 - バージョン 10.0.0 の新機能, 145, 150
- DEBUG 接続パラメータ
 - バージョン 8.0.1 の廃止予定機能, 356
- DECLARE LOCAL TEMPORARY TABLE 文
 - バージョン 11 へのアップグレードのトラブルシューティング, 404
 - バージョン 9.0.1 での動作の変更, 302
- DECOMPRESS 関数

バージョン 10.0.0 の強化, 151
バージョン 9.0.1 の新機能, 287
DECRYPT 関数
 バージョン 11.0.0 の強化, 34
 バージョン 9.0.1 の新機能, 287
dedicated_task オプション
 バージョン 9.0.1 の新機能, 291
default_dbpace オプション
 バージョン 10.0.0 の新機能, 140
default_dbpace プロパティ
 バージョン 10.0.0 の新機能, 143
default_isql_encoding オプション
 バージョン 9.0.1 の新機能, 291
default_timestamp_increment オプション
 バージョン 10.0.1 での動作の変更, 110
DefaultNcharCollation プロパティ
 バージョン 10.0.0 の新機能, 145
DEFAULT VALUES 句
 バージョン 11.0.1 の新機能, 4
delete_old_logs オプション
 バージョン 10.0.1 の強化, 116
DELETE 文
 バージョン 10.0.0 の強化, 154
 バージョン 10.0.1 の強化, 105, 108
 バージョン 11.0.0 での動作の変更, 66
 バージョン 11.0.0 の強化, 48, 55
 バージョン 11.0.1 の強化, 6
demo.db ファイル
 バージョン 10.0.0 でのサンプル・データベース
 の変更, 257
 バージョン 11.0.0 の強化, 99
DENSE_RANK 関数
 バージョン 9.0.1 の新機能, 287
Deployment ウィザード
 バージョン 10.0.0 の新機能, 158
 バージョン 11.0.0 の強化, 92
describe_java_format オプション
 バージョン 10.0.0 でサポート対象外, 169
DESCRIBE 文
 バージョン 10.0.0 の強化, 250
 バージョン 10.0.1 での動作の変更, 111
 バージョン 11.0.0 の強化, 94
DETACH TRACING 文
 バージョン 10.0.0 の新機能, 152
DISH サービス
 バージョン 10.0.1 での動作の変更, 111
DiskReadEng プロパティ
 バージョン 10.0.0 の新機能, 145
 バージョン 11.0.0 での名前の変更, 67
DiskReadHintPages プロパティ
 バージョン 11.0.0 の新機能, 67
DiskReadHintScatterLimit プロパティ
 バージョン 11.0.0 の新機能, 67
DiskReadHint プロパティ
 バージョン 11.0.0 の新機能, 67
DiskReadWorkTable プロパティ
 バージョン 11.0.0 の新機能, 39, 41
DiskRead プロパティ
 バージョン 11.0.0 の新機能, 67
DiskRetryReadScatter
 バージョン 11.0.0 の新機能, 40
DiskRetryReadScatter プロパティ
 バージョン 11.0.0 の新機能, 41
DiskRetryRead プロパティ
 バージョン 11.0.0 の新機能, 40
DiskRetryWrite プロパティ
 バージョン 11.0.0 の新機能, 40
DiskSyncRead プロパティ
 バージョン 11.0.0 の新機能, 39, 41
DiskSyncWrite プロパティ
 バージョン 11.0.0 の新機能, 39, 41
DiskWaitRead プロパティ
 バージョン 11.0.0 の新機能, 39
DiskWaitWrite プロパティ
 バージョン 11.0.0 の新機能, 39
DiskWriteHintPages プロパティ
 バージョン 11.0.0 の新機能, 39, 41
DiskWriteHint プロパティ
 バージョン 11.0.0 の新機能, 39, 41
DISTINCT 句
 バージョン 10.0.0 での Ultra Light の強化, 238
divide_by_zero_error オプション
 バージョン 11.0.0 でサポート終了, 69
DLL プロトコル・オプション
 10.0.0 での動作の変更, 198
 バージョン 11.0.0 でサポート終了, 71
DocCommentXchange (DCX)
 説明, viii
 バージョン 11.0.1 の新機能, 25
Driver 接続パラメータ
 バージョン 11.0.0 の強化, 50
DriveType プロパティ
 バージョン 9.0.1 の新機能, 293
DROP DBSPACE 文

バージョン 10.0.1 での動作の変更, 110

DROP EVENT 文

- バージョン 11.0.0 の強化, 47
- バージョン 11.0.1 の強化, 5

DROP FUNCTION 文

- バージョン 11.0.1 の強化, 5

DROP INDEX 文

- 9.0.1 の Ultra Light 動的 SQL の新機能, 297

DROP LOGIN POLICY 文

- バージョン 11.0.0 の新機能, 45

DROP MATERIALIZED VIEW 文

- バージョン 11.0.1 の強化, 5

DROP PROCEDURE 文

- バージョン 11.0.1 の強化, 5

DROP PUBLICATION 文

- バージョン 10.0.0 での Ultra Light の強化, 238

DROP TABLE 文

- 9.0.1 の Ultra Light 動的 SQL の新機能, 297
- バージョン 11.0.1 の強化, 5

DROP TEXT CONFIGURATION 文

- バージョン 11.0.0 の新機能, 46

DROP TEXT INDEX 文

- バージョン 11.0.0 の新機能, 46

DROP TRIGGER 文

- バージョン 11.0.1 の強化, 5

DROP USER 文

- バージョン 11.0.0 の新機能, 45

DROP VARIABLE 文

- バージョン 11.0.1 の強化, 5

DROP VIEW 文

- バージョン 11.0.1 の強化, 5

DROP 文

- バージョン 10.0.0 の強化, 152

DSN 接続パラメータ

- バージョン 10.0.0 での動作の変更, 169

E

EBF

- データベース・ミラーリング使用時の適用, 407

ECC

- バージョン 10.0.0 で HTTPS に使用可能, 211
- バージョン 10.0.0 の Ultra Light の新機能, 240

ecc_tls

- バージョン 10.0.0 で名前が変更された Mobile Link [mlsrv10] のオプション, 217

Elevate 接続パラメータ

- バージョン 11.0.0 の新機能, 33

Embedded SQL

- Ultra Light 9.0.2 で廃止予定, 281

Embedded SQL のインポート・ライブラリ

- 11.0.0 の新機能, 62

ENAME プロトコル・オプション

- バージョン 11.0.0 でサポート終了, 69

encrypt_aes_random_iv オプション

- バージョン 10.0.1 の新機能, 109
- バージョン 11.0.0 でサポート終了, 71

ENCRYPT_PASSWORD 接続プロパティ

- バージョン 11.0.0 の新機能, 35

EncryptionScope プロパティ

- バージョン 10.0.0 の新機能, 146

Encryption プロパティ

- バージョン 10.0.0 での動作の変更, 168

ENCRYPT 関数

- バージョン 11.0.0 の強化, 34
- バージョン 9.0.1 の新機能, 287

end スクリプト

- バージョン 10.0.0 での Mobile Link の動作の変更, 215

environment.plist

- バージョン 11.0.1 での動作の変更, 12

error_handler

- バージョン 10.0.0 の新機能, 213

ER (実体関連) タブ

- バージョン 10.0.0 の新機能, 249

EVENT_PARAMETER 関数

- バージョン 10.0.0 の強化, 130
- バージョン 11.0.0 の強化, 43

EventTypeDesc

- バージョン 11.0.0 の新機能, 40

EventTypeName

- バージョン 11.0.0 の新機能, 40

example_upload_cursor

- バージョン 10.0.0 で削除, 216

example_upload_delete

- バージョン 10.0.0 で削除, 216

example_upload_insert

- バージョン 10.0.0 で削除, 216

example_upload_update

- バージョン 10.0.0 で削除, 216

EXCEL フォーマット、INPUT 文

- バージョン 11.0.0 で削除されたサポート, 95

EXCEL フォーマット、OUTPUT 文

- バージョン 11.0.0 で削除されたサポート, 95

ExceptionHandler2 デリゲート [QA .NET API]
バージョン 10.0.1 の QAnywhere の新機能, 118
ExceptionHandler デリゲート [QA .NET API]
バージョン 10.0.1 の QAnywhere の新機能, 118
EXCEPT 文
バージョン 10.0.1 の強化, 105
バージョン 11.0.0 での動作の変更, 66
バージョン 11.0.0 の強化, 48
ExchangeTasksCompleted プロパティ
バージョン 10.0.1 の新機能, 109
ExchangeTasks プロパティ
バージョン 10.0.0 の新機能, 145
EXECUTE IMMEDIATE 文
バージョン 9.0.1 での動作の変更, 303
EXIT 文
バージョン 10.0.0 での動作の変更, 251
ExprCacheAbandons プロパティ
バージョン 10.0.0 の新機能, 143
ExprCacheDropsToReadOnly プロパティ
バージョン 10.0.0 の新機能, 143
ExprCacheEvicts プロパティ
バージョン 10.0.0 の新機能, 143
ExprCacheHits プロパティ
バージョン 10.0.0 の新機能, 143
ExprCacheInserts プロパティ
バージョン 10.0.0 の新機能, 143
ExprCacheLookups プロパティ
バージョン 10.0.0 の新機能, 143
ExprCacheResumesOfReadWrite プロパティ
バージョン 10.0.0 の新機能, 143
ExprStarts プロパティ
バージョン 10.0.0 の新機能, 143
ExtendedName プロトコル・オプション
バージョン 11.0.0 でサポート終了, 69

F

FileSize プロパティ
バージョン 10.0.0 で削除された writefile
dbspace サポート, 196
FileVersion プロパティ
バージョン 10.0.0 で廃止予定のデータベース・
プロパティ, 199
FIPS
バージョン 10.0.0 での動作の変更, 168
バージョン 10.0.0 の Ultra Light の新機能, 235
バージョン 10.0.0 の強化, 136

バージョン 10.0.1 での Ultra Light の動作の変更, 122
バージョン 11.0.0 の強化, 34
FIPS オプション
10.0.0 での Mobile Link [mluser] の新機能, 211
バージョン 10.0.0 の Mobile Link [mlsrv10] の新
機能, 211
FIRST_VALUE 関数
バージョン 10.0.1 の新機能, 108
バージョン 11.0.1 の新機能, 4
FirstOption プロパティ
バージョン 10.0.0 の新機能, 145
float_as_double オプション
バージョン 11.0.0 でサポート終了, 69
FORCE NO OPTIMIZATION 句
バージョン 11.0.1 の新機能, 6
FORMAT 句
FORMAT ASCII、バージョン 11.0.0 で廃止予
定, 68
FOR OLAP WORKLOAD オプション
バージョン 10.0.0 の新機能, 155
FOXPRO フォーマット、INPUT 文
バージョン 11.0.0 で削除されたサポート, 95
FOXPRO フォーマット、OUTPUT 文
バージョン 11.0.0 で削除されたサポート, 95
FreePageBitMaps プロパティ
バージョン 10.0.0 で廃止予定のデータベース・
プロパティ, 199
FreePages プロパティ
バージョン 10.0.0 で削除された writefile
dbspace サポート, 196
FROM 句
バージョン 11.0.0 の強化, 29
バージョン 9.0.1 の強化, 289
FunctionMaxParms プロパティ
バージョン 10.0.0 の新機能, 145
FunctionMinParms プロパティ
バージョン 10.0.0 の新機能, 145
FunctionName プロパティ
バージョン 10.0.0 の新機能, 145

G

gencert ユーティリティ
バージョン 10.0.1 で廃止予定, 124
バージョン 11.0.0 で削除, 99
GET_BIT 関数
バージョン 10.0.0 の新機能, 149

GetData プロパティ
バージョン 10.0.0 の新機能, 143
getLastDownloadTime
バージョン 11.0.1 の強化, 15
getPageReads() メソッド
バージョン 11.0.1 での Ultra Light J の強化, 20
getPageWrites メソッド
バージョン 11.0.1 での Ultra Light J の強化, 20
getPropertyNames
バージョン 10.0.0 で削除された QAnywhere C+
+ 関数, 229
getStreamErrorParameters メソッド
バージョン 11.0.1 での Ultra Light の強化, 18
Government Services Edition
Certicom Security Builder、バージョン 10.0.1 の
強化, 124
GROUPING SETS 演算
バージョン 9.0.1 の新機能, 286
GUID 識別子
Ultra Light バージョン 11.0.0 での強化, 83
gzip アルゴリズム
バージョン 10.0.0 の新機能, 151

H

handle_error
バージョン 10.0.0 での動作の変更, 215
handle_odbc_error
バージョン 10.0.0 での動作の変更, 215
HasCollationTailoring プロパティ
バージョン 10.0.1 の新機能, 106
HasEndianSwapFix プロパティ
バージョン 11.0.0 の新機能, 41
HASH 関数
バージョン 10.0.0 の強化, 151
バージョン 9.0.1 の新機能, 287
HEADER 句
バージョン 10.0.0 の新機能, 161
バージョン 10.0.1 の強化, 106
HeapsCarver プロパティ
バージョン 10.0.0 の新機能, 143, 145
HeapsLocked プロパティ
バージョン 10.0.0 の新機能, 143, 145
HeapsQuery プロパティ
バージョン 10.0.0 の新機能, 143, 145
HeapsRelocatable プロパティ
バージョン 10.0.0 の新機能, 145
HistogramHashFix プロパティ

バージョン 10.0.0 で削除されたデータベース・
プロパティ, 199
Histograms プロパティ
バージョン 10.0.0 で削除されたデータベース・
プロパティ, 199
HP-UX
バージョン 10.0.1 の強化, 112
HTML_DECODE 関数
バージョン 10.0.1 での動作の変更, 105
バージョン 10.0.1 の強化, 105
HTTP
バージョン 11.0.0 での動作の変更, 64
http_session_timeout オプション
バージョン 10.0.0 の新機能, 140
http_session_timeout プロパティ
バージョン 10.0.0 の新機能, 143, 146
HttpAddresses プロパティ
バージョン 11.0.0 の新機能, 40
HttpNumActiveReq プロパティ
バージョン 11.0.0 の新機能, 40
HttpNumConnections プロパティ
バージョン 11.0.0 の新機能, 40
HttpNumSessions プロパティ
バージョン 11.0.0 の新機能, 40
HTTPS
バージョン 10.0.1 の強化, 107
バージョン 11.0.0 の強化, 54
https_fips
バージョン 10.0.0 で名前が変更された Mobile
Link [mlsrv10] のオプション, 217
HttpsAddresses プロパティ
バージョン 11.0.0 の新機能, 40
HttpServiceName プロパティ
バージョン 10.0.0 の新機能, 143
HttpsNumActiveReq プロパティ
バージョン 11.0.0 の新機能, 40
HttpsNumConnections プロパティ
バージョン 11.0.0 の新機能, 40
HTTP クライアントのチャンク・モードの転送コー
ディング
11.0.1 での動作の変更, 13
HTTP ヘッダの修正
バージョン 10.0.1 の強化, 106

I

iAnywhere.UltraLite ネームスペース
, 245

-
- iAnywhere JDBC ドライバ
 - バージョン 10.0.0 の強化, 156
 - バージョン 10.0.1 の強化, 112
 - iAnywhere Solutions Oracle ドライバ
 - バージョン 10.0.1 の強化, 109
 - バージョン 11.0.0 の強化, 73
 - iAnywhere デベロッパー・コミュニティ
 - ニュースグループ, xiv
 - ias_CurrentDayOfMonth
 - バージョン 10.0.0 で削除された QAnywhere 転送ルール変数, 229
 - ias_CurrentDayOfWeek
 - バージョン 10.0.0 で削除された QAnywhere 転送ルール変数, 229
 - ias_CurrentMonth
 - バージョン 10.0.0 で削除された QAnywhere 転送ルール変数, 229
 - ias_CurrentYear
 - バージョン 10.0.0 で削除された QAnywhere 転送ルール変数, 229
 - ias_MaxDeliveryAttempts
 - バージョン 10.0.0 の新しい QAnywhere プロパティ, 226
 - ias_MaxUploadSize
 - バージョン 10.0.0 の新しい QAnywhere プロパティ, 226
 - ias_Status
 - バージョン 10.0.0 の新しい QAnywhere プロパティ, 226
 - ias_StatusTime
 - バージョン 10.0.0 の新しい QAnywhere プロパティ, 226
 - IBM DB2
 - バージョン 10.0.0 での ODBC ドライバの動作の変更, 258
 - バージョン 9.0.1 での設定スクリプト名の変更, 295
 - IBM DB2 8.2 CLI ドライバ
 - バージョン 10.0.0 でのサポート, 258
 - ICU
 - バージョン 10.0.0 の新機能, 131
 - IdleTimeout プロパティ
 - 10.0.0 の新機能, 150
 - ID ファイル
 - バージョン 11.0.0 での動作の変更, 100
 - IF EXISTS 句
 - バージョン 11.0.1 の新機能, 5
 - IF NOT EXISTS 句
 - バージョン 11.0.1 の新機能, 5
 - IF 式
 - バージョン 11.0.0 の強化, 47
 - IF 文
 - Ultra Light バージョン 11.0.0 での強化, 85
 - バージョン 11.0.0 の強化, 47
 - ignored_deletes
 - バージョン 10.0.0 の新しい動作, 218
 - ignored_inserts
 - バージョン 10.0.0 の新しい動作, 218
 - ignored_updates
 - バージョン 10.0.0 の新しい動作, 218
 - ignore プロトコル・オプション
 - バージョン 10.0.0 での Mobile Link の動作の変更, 204
 - INDEX ONLY 句
 - バージョン 11.0.0 の新機能, 47
 - IndexStatistics プロパティ
 - バージョン 10.0.0 で削除されたデータベース・プロパティ, 199
 - InfoMaker
 - バージョン 10.0.1 での Vista のサポート, 126
 - Input Method Editor
 - バージョン 9.0.1 での動作の変更, 302
 - INPUT 文
 - バージョン 11.0.0 の強化, 94
 - バージョン 9.0.1 の強化, 291
 - INPUT INTO 文
 - バージョン 11.0.0 での動作の変更, 94
 - INSERT 文
 - バージョン 10.0.0 での動作の変更, 192
 - バージョン 10.0.0 の強化, 154
 - バージョン 10.0.1 の強化, 105, 108
 - バージョン 11.0.0 での動作の変更, 66
 - バージョン 11.0.0 の強化, 48, 55
 - バージョン 11.0.1 の強化, 4, 6
 - install-dir
 - マニュアルの使用方法, xi
 - InstallShield
 - バージョン 10.0.0 での動作の変更, 158
 - InstallShield プロジェクト
 - バージョン 9.0.1 の新機能, 290
 - INSTEAD OF トリガ
 - バージョン 10.0.1 の新機能, 108
 - Intel x86 のプロセッサ
-

- SQL Anywhere 10.0.0 でサポートされるバージョン, 165
- Interactive SQL
 - バージョン 10.0.0 での動作の変更, 251
 - バージョン 10.0.0 の Ultra Light の新機能, 236
 - バージョン 10.0.0 の強化, 249, 250
 - バージョン 11.0.0 の強化, 93
 - バージョン 11.0.0 の新機能, 90
 - バージョン 9.0.1 の強化, 292
- Interactive SQL の新機能
 - バージョン 10.0.0, 248
 - バージョン 11.0.0, 88
 - バージョン 11.0.1, 22
- Interactive SQL の動作の変更
 - バージョン 10.0.0, 250
 - バージョン 11.0.0, 91
 - バージョン 11.0.1, 24
- Interactive SQL ユーティリティ [dbisql]
 - バージョン 10.0.0 の強化, 250
- INTERSECT 文
 - バージョン 10.0.1 の強化, 105
 - バージョン 11.0.0 での動作の変更, 66
 - バージョン 11.0.0 の強化, 48
- INTO LOCAL TEMPORARY TABLE 句
 - バージョン 11.0.1 の新機能, 5
- invalid_extensions オプション
 - バージョン 10.0.0 の新機能, 231
- IN リスト・アルゴリズム (IN)
 - バージョン 11.0.0 の強化, 54
- IOParallelism プロパティ
 - バージョン 10.0.0 の新機能, 146
- IPv6
 - バージョン 10.0.1 の強化, 108
- IPv6 のサポート
 - バージョン 10.0.0 の Ultra Light の新機能, 240
 - バージョン 10.0.0 の新機能, 134
- IPX プロトコル
 - バージョン 8.0.0 ではサポート終了, 386
- IRDK
 - バージョン 11.0.0 での動作の変更, 67
- IsDebugger プロパティ
 - 接続プロパティ、11.0.1 の新機能, 2
- IsEccAvailable プロパティ
 - バージョン 10.0.0 の新機能, 145
- IsJavaAvailable プロパティ
 - バージョン 10.0.0 でサポート対象外, 170
- isolation_level オプション
 - バージョン 10.0.0 の強化, 130
 - バージョン 10.0.1 の強化, 105
- isql_allow_client_file_read
 - バージョン 11.0.0 での名前の変更, 95
- isql_allow_client_file_write
 - バージョン 11.0.0 での名前の変更, 95
- isql_allow_read_client_file オプション
 - バージョン 11.0.0 での新しい名前, 95
- isql_allow_write_client_file オプション
 - バージョン 11.0.0 での新しい名前, 95
- isql_maximum_displayed_rows オプション
 - バージョン 10.0.0 の新機能, 250
- isql_plan オプション
 - バージョン 10.0.0 での動作の変更, 251
 - バージョン 11.0.0 で削除, 96
- isql_show_multiple_result_sets オプション
 - バージョン 10.0.0 の新機能, 250
- IsRsaAvailable プロパティ
 - バージョン 10.0.0 の新機能, 145
- ISYSCAPABILITYNAME
 - バージョン 11.0.0 での動作の変更, 58
- ISYSDBFILE
 - バージョン 11.0.0 での動作の変更, 58
- ISYSDBSPACE
 - バージョン 11.0.0 での動作の変更, 58
- ISYSDBSPACEPERM
 - バージョン 11.0.0 での動作の変更, 58
- ISYSEVENTTYPE
 - バージョン 11.0.0 での動作の変更, 58
- ISYSFILE
 - バージョン 11.0.0 での動作の変更, 58
- ISYSFKEY システム・テーブル
 - バージョン 10.0.0 の新機能のシステム・テーブル, 162
- ISYSIDX
 - バージョン 11.0.0 での動作の変更, 58
- ISYSIDXCOL システム・テーブル
 - バージョン 10.0.0 の新機能のシステム・テーブル, 162
- ISYSLOGINMAP
 - バージョン 11.0.0 の新機能, 58
- ISYSLOGINPOLICY
 - バージョン 11.0.0 の新機能, 58
- ISYSLOGINPOLICYOPTION
 - バージョン 11.0.0 の新機能, 58
- ISYSOBJECT
 - バージョン 11.0.0 での動作の変更, 58

ISYSPHYSIDX

バージョン 10.0.0 の新機能のシステム・テーブル, 162

ISYSTAB

バージョン 11.0.0 での動作の変更, 58

ISYTEXTCONFIG

バージョン 11.0.0 の新機能, 58

ISYTEXTIDX

バージョン 11.0.0 の新機能, 58

ISYTEXTIDXTAB

バージョン 11.0.0 の新機能, 58

ISYSVIEW

バージョン 11.0.0 での動作の変更, 58

Itanium

バージョン 11.0.0 での動作の変更, 66

J

Java

バージョン 10.0.0 での動作の変更, 169

java_heap_size オプション

バージョン 10.0.0 でサポート対象外, 169

java_input_output オプション

バージョン 10.0.0 でサポート対象外, 169

java_input_output プロパティ

バージョン 10.0.0 でサポート対象外, 170

java_location オプション

バージョン 10.0.0 の Mobile Link の新機能, 205

バージョン 10.0.0 の新機能, 141

java_location プロパティ

バージョン 10.0.0 の新機能, 143

java_main_userid オプション

バージョン 10.0.0 の新機能, 141

java_main_userid プロパティ

バージョン 10.0.0 の新機能, 143

java_namespace_size オプション

バージョン 10.0.0 でサポート対象外, 169

java_page_buffer_size オプション

バージョン 10.0.0 でサポート対象外, 169

java_vm_options オプション

バージョン 10.0.0 の新機能, 141

Java API

バージョン 10.0.0 の QAnywhere の新機能, 226

Java DownloadTableData インタフェース

バージョン 11.0.0 の強化, 74

JavaGlobFix プロパティ

バージョン 10.0.0 でサポート対象外, 170

JavaHeapSize プロパティ

バージョン 10.0.0 でサポート対象外, 170

Java ME

Ultra Light バージョン 11.0.0 での強化, 86

JavaNSSize プロパティ

バージョン 10.0.0 でサポート対象外, 170

Java SE

Ultra Light バージョン 11.0.0 での強化, 86

JavaVM プロパティ

バージョン 10.0.0 の新機能, 146

Java VM プロパティ

バージョン 10.0.0 の新機能, 170

jConnect

バージョン 10.0.0 で Interactive SQL への接続に使用できない, 252

バージョン 10.0.0 で SQL Anywhere コンソール (dbconsole) への接続に使用できない, 252

バージョン 10.0.0 で Sybase Central への接続に使用できない, 252

バージョン 10.0.0 で削除されたバージョン 4.5 のサポート, 193

バージョン 10.0.0 の強化, 165

バージョン 10.0.1 での動作の変更, 112

バージョン 11.0.0 の強化, 35

JDBC 2.0

サポート終了, 156

JDBC 3.0

バージョン 10.0.0 の新機能, 156

JDKVersion プロパティ

バージョン 10.0.0 でサポート対象外, 170

K

keep-alive request-header フィールド

バージョン 10.0.0 の新機能, 160

KeepaliveTimeout プロトコル・オプション

バージョン 10.0.0 の新機能, 160

Kerberos

バージョン 10.0.0 の新機能, 136

KTO プロトコル・オプション

バージョン 10.0.0 の新機能, 160

Kyocera

バージョン 10.0.0 で Mobile Link Palm Listener でのサポート廃止, 224

L

LargeProcedureIDs プロパティ

バージョン 10.0.0 で削除されたデータベース・プロパティ, 199

- LAST_BACKUP 操作
 - バージョン 10.0.0 の新機能, 135
 - LAST_VALUE 関数
 - バージョン 10.0.1 の新機能, 108
 - バージョン 11.0.1 の新機能, 4
 - LastConnectionProperty プロパティ
 - バージョン 10.0.0 の新機能, 145
 - LastDatabaseProperty プロパティ
 - バージョン 10.0.0 の新機能, 145
 - LastOption プロパティ
 - バージョン 10.0.0 の新機能, 145
 - LastPlanText プロパティ
 - バージョン 10.0.0 の新機能, 143
 - LastServerProperty プロパティ
 - バージョン 10.0.0 の新機能, 145
 - LastStatement プロパティ
 - バージョン 10.0.1 での動作の変更, 111
 - LazyClose 接続パラメータ
 - バージョン 11.0.0 での動作の変更, 62
 - LCLOSE 接続パラメータ
 - バージョン 11.0.0 での動作の変更, 62
 - LDAP 認証
 - 9.0.1 の Mobile Link の強化, 294
 - バージョン 10.0.0 の強化, 135
 - LENGTH 関数
 - バージョン 10.0.0 の QAnywhere の新機能, 228
 - libdbtool9.so
 - バージョン 9.0.2 で廃止予定, 277
 - LicensesInUse プロパティ
 - バージョン 10.0.0 の名前の変更, 168
 - Linux
 - バージョン 10.0.0 の新機能, 159
 - バージョン 11.0.0 での動作の変更, 66
 - バージョン 11.0.0 の強化, 53
 - バージョン 11.0.0 の新機能, 53
 - バージョン 9.0.1 の新機能, 293
 - liveness_timeout オプション
 - バージョン 10.0./0 で置き換えられた Mobile Link クライアント・プロトコル・オプション, 209
 - バージョン 10.0.0 で削除, 217
 - LOAD TABLE 文
 - バージョン 10.0.0 での動作の変更, 193
 - バージョン 10.0.0 の強化, 153
 - バージョン 11.0.0 の強化, 30, 44, 45, 47, 75, 80
 - バージョン 9.0.1 の強化, 287
 - バージョン 9.0.1 の変数の強化サポート, 287
 - LockCount プロパティ
 - バージョン 10.0.0 の新機能, 143, 146
 - LockedCursorPages プロパティ
 - バージョン 10.0.0 の新機能, 143
 - LockIndexID プロパティ
 - バージョン 11.0.0 の新機能, 39
 - LockRowID プロパティ
 - バージョン 11.0.0 の新機能, 39
 - LockTableOID プロパティ
 - バージョン 10.0.0 の新機能, 143
 - login_mode オプション
 - バージョン 10.0.0 での動作の変更, 175
 - login_procedure オプション
 - バージョン 11.0.0 の強化, 37
 - LogMaxSize プロトコル・オプション
 - バージョン 10.0.0 の強化, 134
 - Log Transfer Manager ユーティリティ [dbltm]
 - バージョン 10.0.0 の強化, 138
 - LONG VARBIT データ型
 - バージョン 10.0.0 の新機能, 155
 - LOTUS フォーマット、INPUT 文
 - バージョン 11.0.0 で削除されたサポート, 95
 - LOTUS フォーマット、OUTPUT 文
 - バージョン 11.0.0 で削除されたサポート, 95
 - LTM
 - バージョン 10.0.0 の強化, 138
- ## M
- MachineName プロパティ
 - バージョン 10.0.1 の強化, 112
 - Mac OS X
 - 11.0.1 での動作の変更, 12
 - SQL Anywhere 11 用にデータベースを再構築, 393
 - バージョン 10.0.1 の強化, 103
 - バージョン 11.0.0 の強化, 53
 - バージョン 11.0.1 での動作の変更, 12
 - バージョン 11.0.1 の強化, 8
 - MAPI メッセージ・タイプ
 - SQL Remote バージョン 10.0.1 でサポート廃止予定, 120
 - バージョン 11.0.0 でサポート終了, 80
 - MapPages プロパティ
 - バージョン 11.0.0 でサポート終了, 68
 - MapPhysicalMemoryEng プロパティ
 - バージョン 10.0.0 の新機能, 145
 - materialized_view_optimization オプション

バージョン 10.0.0 の新機能, 140
materialized_view_optimization プロパティ
バージョン 10.0.0 の新機能, 143
max_client_statements_cached オプション
バージョン 10.0.1 の新機能, 104
max_client_statements_cached プロパティ
バージョン 10.0.1 の新機能, 104
max_hash_size オプション
バージョン 10.0.0 でサポート対象外, 194
max_priority オプション
バージョン 11.0.0 の新機能, 37
max_priority プロパティ
バージョン 11.0.0 の新機能, 39
max_query_tasks オプション
バージョン 10.0.1 の強化, 105
max_query_tasks プロパティ
バージョン 10.0.0 の新機能, 143
max_temp_space オプション
バージョン 10.0.0 の新しい接続プロパティ,
143
バージョン 10.0.0 の新機能, 140
max_work_table_hash_size オプション
バージョン 10.0.0 でサポート対象外, 194
MaxConnections プロパティ
バージョン 10.0.0 の新機能, 145
MaxEventType
バージョン 11.0.0 の新機能, 40
MaxMessage プロパティ
バージョン 11.0.0 で廃止予定, 69
MaxRemoteCapability プロパティ
バージョン 11.0.0 の新機能, 40
MaxRequestSize プロトコル・オプション
バージョン 10.0.0 の強化, 134
MDSR 暗号化
バージョン 9.0.1 でサポート終了, 301
MERGE 文
バージョン 11.0.0 の新機能, 29
MessageCategoryLimit プロパティ
バージョン 11.0.0 の新機能, 40
MessageText プロパティ
バージョン 11.0.0 で廃止予定, 69
MessageTime プロパティ
バージョン 11.0.0 で廃止予定, 69
MessageWindowSize プロパティ
バージョン 11.0.0 で廃止予定, 69
Message プロパティ
バージョン 11.0.0 で廃止予定, 69

MESSAGE 文

バージョン 10.0.0 の強化, 154

バージョン 9.0.1 の強化, 288

Microsoft Access

バージョン 11.0.0 でのリモート・データ・アク
セスの強化, 55

Microsoft Excel

バージョン 11.0.0 で削除された INPUT 文のサ
ポート, 95

バージョン 11.0.0 で削除された OUTPUT 文の
サポート, 95

Microsoft SQL Server 2008 のサポート

バージョン 11.0.1 の強化, 14

Microsoft 分散トランザクション・コーディネータ

バージョン 11.0.0 の新機能, 73

Microsoft ボリューム・シャドウ・コピー・サービ ス (VSS)

バージョン 11.0.0 の新機能, 33

min_table_size_for_histogram オプション

バージョン 9.0.1 での動作の変更, 302

バージョン 9.0.2 で廃止予定, 277

MIPS

バージョン 10.0.0 で削除された SQL Anywhere
サポート, 200

MirrorFailover システム・イベント

バージョン 10.0.0 の新機能, 129

MirrorMode プロパティ

バージョン 11.0.0 の新機能, 41

MirrorServerDisconnect システム・イベント

バージョン 10.0.0 の新機能, 129

MirrorState プロパティ

バージョン 10.0.0 の新機能, 146

ml_add_colum システム・プロシージャ

バージョン 10.0.0 の新機能, 204

ml_column

バージョン 10.0.0 の新機能, 203

ml_database

バージョン 10.0.0 の新機能, 203

ml_delete_sync_state_before システム・プロシ ージャ

バージョン 10.0.0 の Mobile Link の新機能, 203

ml_delete_sync_state システム・プロシージャ

バージョン 10.0.0 の Mobile Link の新機能, 203

ML_GET_SERVER_NOTIFICATION 関数

バージョン 11.0.1 での Ultra Light の強化, 18

ml_global スクリプト・バージョン

バージョン 10.0.0 の Mobile Link の新機能, 206

- ml_ignore
 - バージョン 11.0.1 の MobiLink (mlsrv11) の新機能, 14
- ml_listening
 - バージョン 10.0.0 の新しいスキーマ, 203
- ml_qa_clients
 - バージョン 10.0.0 の新機能, 203
- ml_qa_delivery
 - バージョン 10.0.0 の新しいスキーマ, 204
- ml_qa_delivery_client
 - バージョン 10.0.0 の新しいスキーマ, 204
- ml_qa_global_props
 - バージョン 10.0.0 での変更, 204
 - バージョン 10.0.0 の新しいスキーマ, 204
- ml_qa_global_props_client
 - バージョン 10.0.0 での変更, 204
- ml_qa_repository_client
 - バージョン 10.0.0 での変更, 204
- ml_qa_repository_props
 - バージョン 10.0.0 での変更, 204
- ml_qa_repository_props_client
 - バージョン 10.0.0 での変更, 204
- ml_qa_repository_staging
 - バージョン 10.0.0 での変更, 204
- ml_remote_id
 - バージョン 11.0.1 での Ultra Light の強化, 18
- ml_remote_id オプション
 - バージョン 10.0.0 の新機能, 208
- ml_reset_sync_state システム・プロシージャ
 - バージョン 10.0.0 の Mobile Link の新機能, 203
- ml_script
 - バージョン 10.0.0 の新しいスキーマ, 203
- ml_subscription
 - バージョン 10.0.0 の新しいスキーマ, 203
- ml_user
 - バージョン 10.0.0 の新しいスキーマ, 203
- mlsrv10.lic
 - バージョン 10.0.1 の新機能, 125
- mlxtract
 - バージョン 10.0.0 での削除, 222
- Mobile Link
 - アップグレード, 410
- Mobile Link ASA クライアントのローワイズ分割
 - 9.0.1 の Mobile Link の新しい動作, 303
- Mobile Link クライアント
 - バージョン 10.0.0 で削除された 9.0 以前へのサポート, 223
- Mobile サーバ
 - アップグレード, 415
- Mobile Link サーバ・ファーム
 - バージョン 11.0.0 の新機能, 73
- Mobile Link システム・オブジェクト
 - バージョン 11.0.0 の強化, 72
- Mobile Link システム・データベース
 - バージョン 11.0.0 の新機能, 72
- Mobile Link システム・テーブル
 - 9.0.1 の新しいテーブル, 294
 - アップグレード, 411
- Mobile Link システム・プロシージャ
 - バージョン 11.0.0 の強化, 72
- Mobile Link の新機能
 - バージョン 10.0.0, 201
 - バージョン 10.0.1, 115
 - バージョン 11.0.0, 72
 - バージョン 11.0.1, 14
 - バージョン 9.0.2, 267
- Mobile Link の動作の変更
 - バージョン 10.0.0, 214
 - バージョン 10.0.1, 116
 - バージョン 11.0.0, 76
 - バージョン 11.0.1, 16
 - バージョン 9.0.2, 279
- Mobile Link ファイル転送
 - バージョン 10.0.0 での Mobile Link の新機能, 208
- Mobile Link プラグイン
 - バージョン 10.0.0 の強化, 249
- Mobile Link モニタ
 - バージョン 10.0.0 の強化, 206
 - バージョン 11.0.0 での動作の変更, 96
- Mobile Link ユーザ名
 - バージョン 10.0.0 での動作の変更, 221
- Mobile Link リレー・サーバ
 - バージョン 11.0.0 の新機能, 73
- Mobile Link ログ・ファイル
 - Ultra Light バージョン 11.0.0 での強化, 83
- Mobile Link ログ・ファイル・ビューワ
 - バージョン 10.0.1 の新機能, 115
- msaccessodbc サーバ・クラス
 - バージョン 11.0.0 の新機能, 55
- MultiPageAllocs プロパティ
 - バージョン 10.0.0 の新機能, 143
- MultiProgrammingLevel プロパティ
 - バージョン 10.0.0 の新機能, 145

mysqlodbc サーバ・クラス
バージョン 11.0.0 の新機能, 55
MySQL の統合データベース
バージョン 11.0.0 の新機能, 72

N

NamedConstraints プロパティ
バージョン 10.0.0 で削除されたデータベース・
プロパティ, 199
Native Ultra Light for Java API サポート
バージョン 10.0.0 で削除された Ultra Light,
245
NcharCharSet プロパティ
バージョン 10.0.0 の新機能, 146
NcharCollation プロパティ
バージョン 10.0.0 の新機能, 146
NCHAR データ型
バージョン 10.0.0 の新機能, 131
NetBios プロトコル
バージョン 8.0.0 ではサポート終了, 386
network_connect_timeout オプション
バージョン 10.0./0 で置き換えられた Mobile
Link クライアント・プロトコル・オプション,
209
network_name プロトコル・オプション
バージョン 10.0.1 の強化, 116
new_row_cursor
バージョン 10.0.0 で削除, 214
バージョン 9.0.0 で廃止予定, 329
NewPassword 接続パラメータ
バージョン 11.0.0 の新機能, 33
NEWPWD 接続パラメータ
バージョン 11.0.0 の新機能, 33
NEXT_SOAP_HEADER 関数
バージョン 10.0.0 の新機能, 161
NextScheduleTime プロパティ
バージョン 10.0.0 の新機能, 150
non_keywords オプション
バージョン 11.0.0 での動作の変更, 64
NoSyncOnStartup dbmsync 拡張オプション
バージョン 10.0.0 の新機能, 210
NULL
バージョン 9.0.1 での動作の変更, 302
NULL 定数の変換
バージョン 10.0.0 での動作の変更, 173
NumLogicalProcessorsUsed プロパティ
バージョン 10.0.0 の新機能, 145

NumLogicalProcessors プロパティ
バージョン 10.0.0 の新機能, 145
NumPhysicalProcessorsUsed プロパティ
バージョン 10.0.0 の新機能, 145
NumPhysicalProcessors プロパティ
バージョン 10.0.0 の新機能, 145
NumProcessorsAvail プロパティ
バージョン 10.0.0 でサポート対象外, 193
NumProcessorsMax プロパティ
バージョン 10.0.0 でサポート対象外, 193
NVFS
バージョン 10.0.0 での Ultra Light の強化, 237

O

ODBC
9.0.1 の Ultra Light の新機能, 297
バージョン 10.0.0 での動作の変更, 169
odbc_distinguish_char_and_varchar オプション
バージョン 9.0.1 の新機能, 294
ODBC サーバ・クラス
バージョン 11.0.0 の強化, 55
[ODBC 設定] ウィンドウ
バージョン 9.0.1 での動作の変更, 302
ODBC データ・ソースとして保存
バージョン 11.0.0 の新機能, 88
ODBC ドライバ
バージョン 10.0.0 の新しいドライバ, 258
バージョン 10.0.1 の新しい Oracle ドライバ,
123
ODBC ドライバ・マネージャ
バージョン 10.0.0 の UNIX の新機能, 159
バージョン 11.0.0 の強化, 50
oem_string オプション
バージョン 10.0.0 の新機能, 140
oem_string プロパティ
バージョン 10.0.0 の新機能, 143
OEM.ini
バージョン 10.0.0 の新機能, 256
バージョン 11.0.0 の強化, 91
old_row_cursor
バージョン 10.0.0 で削除, 214
バージョン 9.0.0 で廃止予定, 329
OLE DB
バージョン 10.0.0 での動作の変更, 169
Open Client
バージョン 11.0.0 の強化, 35
OpenString アルゴリズム

- バージョン 10.0.0 の新機能, 32
- OPENSTRING 句
 - バージョン 11.0.0 の新機能, 32
- OpenTableEx 関数 [UL C++]
 - Ultra Light バージョン 11.0.0 での強化, 85
- openxml システム・プロシージャ
 - バージョン 10.0.0 での動作の変更, 173
 - バージョン 9.0.1 の強化, 293
- OPEN 文
 - バージョン 10.0.0 の強化, 130
- optimistic_wait_for_commit オプション
 - バージョン 11.0.0 でサポート終了, 69
- optimization_goal オプション
 - バージョン 10.0.1 の強化, 105
- optimization_level オプション
 - バージョン 10.0.1 の強化, 105
- optimization_workload オプション
 - バージョン 10.0.1 の強化, 105
 - バージョン 9.0.1 の新機能, 289
- OptionWatchAction プロパティ
 - バージョン 11.0.0 の新機能, 40
- OptionWatchList プロパティ
 - バージョン 11.0.0 の新機能, 40
- OPTION 句
 - バージョン 10.0.0 の強化, 152
 - バージョン 10.0.1 の強化, 105
 - バージョン 11.0.0 での動作の変更, 66
 - バージョン 11.0.0 の強化, 48
- Oracle
 - バージョン 11.0.0 の強化, 73
- Oracle varray
 - バージョン 11.0.1 の新機能, 14
- Oracle ドライバ
 - バージョン 10.0.1 の強化, 109
 - バージョン 11.0.0 の強化, 73
- ORDER BY 句
 - バージョン 10.0.0 での Ultra Light の強化, 238
- ORDER BY と UPDATE 文
 - バージョン 9.0.1 での動作の変更, 303
- OR REPLACE 句
 - バージョン 11.0.1 の新機能, 5
- OSUser プロパティ
 - バージョン 11.0.0 の新機能, 39
- OUTER APPLY 句
 - バージョン 11.0.0 の新機能, 47
- OUTPUT 文
 - バージョン 11.0.0 での dbisql の動作の変更, 68

- バージョン 11.0.0 での動作の変更, 95
- バージョン 11.0.0 の強化, 94
- バージョン 11.0.1 での動作の変更, 12
- バージョン 9.0.1 の強化, 291

P

- Palm HotSync コンジット・インストーラ・ユーティリティ
 - バージョン 10.0.0 での Ultra Light の強化, 237
- Palm OS
 - 9.0.1 でサポートされる最も古いバージョン, 304
 - Certicom Security Builder GSE のバージョン、バージョン 10.0.1 の強化, 124
 - バージョン 10.0.0 での Ultra Light の強化, 235
- PartnerState プロパティ
 - バージョン 10.0.0 の新機能, 146
- PBUF 接続パラメータ
 - バージョン 11.0.0 の強化, 33
- PDF
 - マニュアル, viii
- percent_as_comment オプション
 - バージョン 11.0.0 での動作の変更, 69
- PERCENT_RANK 関数
 - バージョン 9.0.1 の新機能, 287
- Perl
 - バージョン 9.0.1 の新しい DBD::ASAny ドライバ, 290
- Perl DBD::ASAny
 - バージョン 10.0.0 での名前の変更, 157
- PHP モジュール
 - 関数名の変更、バージョン 11.0.0 での動作の変更, 61
 - バージョン 10.0.0 の強化, 157
 - バージョン 10.0.0 の動作の変更, 171
- ping ユーティリティ [dbping]
 - バージョン 10.0.0 での動作の変更, 170
 - バージョン 10.0.0 の新機能, 138
- PORT プロトコル・オプション
 - バージョン 10.0.0 での動作の変更, 173
- post_login_procedure オプション
 - バージョン 10.0.0 の新機能, 143
 - バージョン 11.0.0 での動作の変更, 64
- PowerDesigner
 - バージョン 10.0.1 での Vista のサポート, 126
- precision オプション
 - バージョン 11.0.0 での動作の変更, 65

PrefetchBuffer 接続パラメータ
バージョン 10.0.0 での動作の変更, 171
バージョン 10.0.0 の強化, 134
バージョン 11.0.0 の強化, 33
PrefetchRows 接続パラメータ
バージョン 10.0.1 での動作の変更, 111
PrefetchRows プロパティ [SA .NET 2.0]
バージョン 10.0.1 での動作の変更, 111
PreparedStatement.addBatch メソッド
バージョン 10.0.0 の新機能, 156
Prepares プロパティ
データベース・プロパティ、11.0.1 の新機能,
2
PreserveSource プロパティ
バージョン 10.0.0 で廃止予定のデータベース・
プロパティ, 199
バージョン 11.0.0 でサポート終了, 68
priority オプション
バージョン 11.0.0 の新機能, 37
priority プロパティ
バージョン 11.0.0 の新機能, 39
procedure_profiling サーバ・オプション
バージョン 10.0.0 での動作の変更, 173
ProfileFilterConn プロパティ
バージョン 10.0.0 の新機能, 150
PROFILE 権限
バージョン 11.0.0 の新機能, 35
PROPERTY_NAME 関数
バージョン 11.0.0 の強化, 41
PROPERTY 関数
バージョン 10.0.0 の強化, 149
Python
Python データベース API、11.0.0 の新機能, 48

Q

QAMessageListener2 インタフェース [QA Java API]
バージョン 10.0.1 の QAnywhere の新機能, 118
QAMessageListener2 デリゲート [QA .NET API]
バージョン 10.0.1 の QAnywhere の新機能, 118
QAnywhere
9.0.1 の Mobile Link の新機能, 294
データベースとアプリケーションのアップグ
レード, 418
QAnywhere .NET API
バージョン 11.0.0 の強化, 78
QAnywhere 11 Demo データ・ソース
バージョン 11.0.0 での動作の変更, 100
QAnywhere Agent
バージョン 10.0.0 での動作の変更, 227
QAnywhere C# API
バージョン 11.0.0 の強化, 78
QAnywhere Java API
バージョン 11.0.0 の強化, 78
QAnywhere Ultra Light Agent
バージョン 11.0.0 の新機能, 78
QAnywhere Web サービス
バージョン 10.0.0 の新機能, 225
QAnywhere サーバ・ログ・ファイル・ビューワ
バージョン 10.0.1 の新機能, 118
QAnywhere スタンドアロン・クライアント
バージョン 11.0.0 の新機能, 17
QAnywhere の新機能
バージョン 10.0.0, 225
バージョン 10.0.1, 118
バージョン 11.0.0, 78
バージョン 11.0.1, 17
バージョン 9.0.2, 273
QAnywhere の動作の変更
バージョン 10.0.0, 228
バージョン 10.0.1, 119
バージョン 11.0.0, 79
バージョン 9.0.2, 282
QAnywhere プラグイン
バージョン 10.0.0 の新機能, 249
qastop ユーティリティ
バージョン 11.0.0 での動作の変更, 79
qauagent ユーティリティ
バージョン 11.0.0 の新機能, 78
query_mem_timeout オプション
バージョン 11.0.0 の新機能, 37
query_mem_timeout プロパティ
バージョン 11.0.0 の新機能, 39
query_plan_on_open オプション
バージョン 11.0.0 でサポート終了, 69
QueryBypassedCosted プロパティ
接続プロパティ、11.0.1 の新機能, 2
データベース・プロパティ、11.0.1 の新機能,
2
QueryBypassedHeuristic プロパティ
接続プロパティ、11.0.1 の新機能, 2
データベース・プロパティ、11.0.1 の新機能,
2
QueryBypassedOptimized プロパティ
接続プロパティ、11.0.1 の新機能, 2

データベース・プロパティ、11.0.1 の新機能, 2

QueryDescribedBypass プロパティ
接続プロパティ、11.0.1 の新機能, 2
データベース・プロパティ、11.0.1 の新機能, 2

QueryDescribedOptimizer プロパティ
接続プロパティ、11.0.1 の新機能, 2
データベース・プロパティ、11.0.1 の新機能, 2

QueryHeapPages プロパティ
バージョン 10.0.0 の新機能, 143, 145

QueryMemActiveCurr プロパティ
バージョン 11.0.0 の新機能, 39, 40

QueryMemActiveEst プロパティ
バージョン 11.0.0 の新機能, 40

QueryMemActiveMax プロパティ
バージョン 11.0.0 の新機能, 40

QueryMemExtraAvail プロパティ
バージョン 11.0.0 の新機能, 39, 40

QueryMemGrantBaseMI プロパティ
バージョン 11.0.0 の新機能, 40

QueryMemGrantBase プロパティ
バージョン 11.0.0 の新機能, 40

QueryMemGrantExtra プロパティ
バージョン 11.0.0 の新機能, 40

QueryMemGrantFailed プロパティ
バージョン 11.0.0 の新機能, 39, 40

QueryMemGrantGranted プロパティ
バージョン 11.0.0 の新機能, 39, 40

QueryMemGrantRequested プロパティ
バージョン 11.0.0 の新機能, 39, 40

QueryMemGrantWaiting プロパティ
バージョン 11.0.0 の新機能, 39, 40

QueryMemPages プロパティ
バージョン 11.0.0 の新機能, 40

QueryMemPercentOfCache プロパティ
バージョン 11.0.0 の新機能, 40

QueryMemWaited プロパティ
バージョン 11.0.0 の新機能, 39, 40

QueryOpened プロパティ
接続プロパティ、11.0.1 の新機能, 2
データベース・プロパティ、11.0.1 の新機能, 2

QueryReused プロパティ
バージョン 11.0.0 の新機能, 41

quoted_identifier オプション

バージョン 10.0.0 での動作変更, 250
バージョン 11.0.0 での動作の変更, 65

R

RAND 関数
バージョン 10.0.0 での動作の変更, 167

RANK 関数
バージョン 9.0.1 の新機能, 287

read_authdn パラメータ
バージョン 10.0.0 の新機能, 135

READ_CLIENT_FILE 関数
バージョン 11.0.0 の新機能, 41

read_password パラメータ
バージョン 10.0.0 の新機能, 135

readcert ユーティリティ
バージョン 10.0.1 で廃止予定, 124
バージョン 11.0.0 で削除, 99

READCLIENTFILE 権限
バージョン 11.0.0 の新機能, 35, 41

READFILE 権限
バージョン 11.0.0 の新機能, 35

ReadHintScatterLimit プロパティ
バージョン 11.0.0 での名前の変更, 67
バージョン 11.0.0 の新機能, 40

ReadHintScatter プロパティ
バージョン 11.0.0 での名前の変更, 67
バージョン 11.0.0 の新機能, 39

ReadHint プロパティ
バージョン 11.0.0 での名前の変更, 67
バージョン 11.0.0 の新機能, 39

READPAST テーブル・ヒント
バージョン 10.0.0 の新機能, 153

READ 文
バージョン 9.0.1 の強化, 291

ReceiveBufferSize プロトコル・オプション
バージョン 10.0.0 の強化, 134

ReceivingTracingFrom プロパティ
バージョン 10.0.0 の新機能, 146

REFRESH MATERIALIZED VIEW 文
バージョン 10.0.0 の新機能, 152
バージョン 11.0.0 での動作の変更, 61
バージョン 11.0.0 の強化, 31

REFRESH TEXT INDEX 文
バージョン 11.0.0 の新機能, 46

REFRESH TRACING LEVEL 文
バージョン 10.0.0 の新機能, 152

REFRESH 句、ALTER MATERIALIZED VIEW 文

バージョン 11.0.0 の新機能, 31
REGBIN プロトコル・オプション
バージョン 11.0.0 でサポート終了, 69
REGEXP_SUBSTR 関数
バージョン 11.0.0 の新機能, 29, 41
バージョン 11.0.1 での動作の変更, 10
REGEXP 探索条件
バージョン 11.0.0 の新機能, 29
バージョン 11.0.1 での動作の変更, 10
RegisterBindery プロトコル・オプション
バージョン 11.0.0 でサポート終了, 69
REGR_AVGX 関数
バージョン 9.0.1 の新機能, 287
REGR_AVGY 関数
バージョン 9.0.1 の新機能, 287
REGR_COUNT 関数
バージョン 9.0.1 の新機能, 287
REGR_INTERCEPT 関数
バージョン 9.0.1 の新機能, 287
REGR_R2 関数
バージョン 9.0.1 の新機能, 287
REGR_SLOPE 関数
バージョン 9.0.1 の新機能, 287
REGR_SXX 関数
バージョン 9.0.1 の新機能, 287
REGR_SXY 関数
バージョン 9.0.1 の新機能, 287
REGR_SYY 関数
バージョン 9.0.1 の新機能, 287
reload.sql
バージョン 11.0.0 の強化, 56
RememberLastPlan プロパティ
バージョン 10.0.0 の新機能, 145
RememberLastStatement プロパティ
バージョン 10.0.0 での動作の変更, 166
バージョン 10.0.1 での動作の変更, 111
RemoteCapability プロパティ
バージョン 11.0.0 の新機能, 40
REMOTE DBA パーミッション
バージョン 10.0.0 での動作の変更, 192
RemoteputWait プロパティ
バージョン 10.0.0 の新機能, 145
REORGANIZE TABLE 文
バージョン 11.0.0 での動作の変更, 62
Replication Agent
バージョン 10.0.0 の強化, 138
ReqCountActive プロパティ
バージョン 10.0.0 の新機能, 143
ReqCountBlockContention プロパティ
バージョン 10.0.0 の新機能, 143
ReqCountBlockIO プロパティ
バージョン 10.0.0 の新機能, 143
ReqCountBlockLock プロパティ
バージョン 10.0.0 の新機能, 143
ReqCountUnscheduled プロパティ
バージョン 10.0.0 の新機能, 143
ReqStatus プロパティ
バージョン 10.0.0 の新機能, 143
ReqTimeActive プロパティ
バージョン 10.0.0 の新機能, 143
ReqTimeBlockContention プロパティ
バージョン 10.0.0 の新機能, 143
ReqTimeBlockIO プロパティ
バージョン 10.0.0 の新機能, 143
ReqTimeBlockLock プロパティ
バージョン 10.0.0 の新機能, 143
ReqTimeUnscheduled プロパティ
バージョン 10.0.0 の新機能, 143
reqtool ユーティリティ
バージョン 10.0.1 で廃止予定, 124
バージョン 11.0.0 で削除, 99
request_timeout オプション
バージョン 10.0.0 の新機能, 140
RequestFilterConn プロパティ
バージョン 10.0.0 の新機能, 145
RequestFilterDB プロパティ
バージョン 10.0.0 の新機能, 145, 150
RequestLogging プロパティ
バージョン 10.0.0 の強化, 150
RequestLogMaxSize プロパティ
バージョン 10.0.0 の新機能, 145
RequestsReceived プロパティ
バージョン 10.0.0 の新機能, 143
RequestTiming プロパティ
バージョン 10.0.0 の新機能, 151
reset.sql
オートインクリメント・カラムのあるテーブル
の再ロード, 397
RESTORE DATABASE 文
バージョン 9.0.1 の変数の強化サポート, 287
RetryConnectionTimeout 接続パラメータ
バージョン 10.0.0 の新機能, 134
RetryConnectionTimeout プロパティ
バージョン 10.0.0 の新機能, 143

return_java_as_string オプション
バージョン 10.0.0 でサポート対象外, 169

REVERSE 関数
バージョン 10.0.0 の新機能, 149

REVOKE CONNECT 文
10.0.0 での動作の変更, 191

ri_trigger_time オプション
バージョン 11.0.0 でサポート終了, 69

RIM
Ultra Light バージョン 11.0.0 での強化, 86

ROLLBACK 文
9.0.1 の Ultra Light 動的 SQL の新機能, 297

ROLLUP 演算
バージョン 9.0.1 の新機能, 286

ROW_NUMBER 関数
バージョン 9.0.1 の新機能, 287

RSA
SQL Anywhere バージョン 10.0.0 に付属, 136
バージョン 10.0.0 の Ultra Light の新機能, 240
バージョン 10.0.0 用の Mobile Link に付属, 211
バージョン 10.0.0 用の Ultra Light に付属, 235
バージョン 10.0.1 の強化, 103

rsa_tls
バージョン 10.0.0 で名前が変更された Mobile Link [mlsrv10] のオプション, 217

rsa_tls_fips
バージョン 10.0.0 で名前が変更された Mobile Link [mlsrv10] のオプション, 217

Ruby
バージョン 11.0.1 の新機能, 6

S

sa_ansi_standard_packages システム・プロシージャ
バージョン 10.0.1 の新機能, 104

sa_char_terms システム・プロシージャ
バージョン 11.0.0 の新機能, 42

sa_check_commit システム・プロシージャ
バージョン 10.0.1 での動作の変更, 110

sa_clean_database システム・プロシージャ
バージョン 10.0.0 の新機能, 147

sa_column_stats システム・プロシージャ
バージョン 10.0.0 の新機能, 147

sa_conn_info システム・プロシージャ
バージョン 10.0.0 の強化, 151
バージョン 10.0.1 での動作の変更, 110

sa_conn_list システム・プロシージャ
バージョン 10.0.0 の新機能, 147

sa_conn_options システム・プロシージャ
バージョン 10.0.0 の新機能, 147

sa_conn_properties_by_conn システム・プロシージャ
バージョン 10.0.0 でサポート対象外, 199

sa_conn_properties_by_name システム・プロシージャ
バージョン 10.0.0 でサポート対象外, 199

sa_conn_properties システム・プロシージャ
バージョン 11.0.0 の強化, 44

sa_convert_ml_progress_to_timestamp システム・プロシージャ
バージョン 10.0.0 の新機能, 210

sa_convert_timestamp_to_ml_progress システム・プロシージャ
バージョン 10.0.0 の新機能, 210

sa_db_list システム・プロシージャ
バージョン 10.0.0 の新機能, 147

sa_db_properties システム・プロシージャ
バージョン 11.0.0 の強化, 44

sa_dependent_views システム・プロシージャ
バージョン 10.0.1 での動作の変更, 110

sa_describe_query システム・プロシージャ
バージョン 10.0.0 の新機能, 147

sa_disk_free_space システム・プロシージャ
バージョン 11.0.0 の強化, 42

sa_external_library_unload システム・プロシージャ
バージョン 11.0.0 の新機能, 43

sa_get_bits システム・プロシージャ
バージョン 10.0.0 の新機能, 147

sa_get_dtt_groupreads システム・プロシージャ
バージョン 11.0.0 の新機能, 41

sa_get_dtt システム・プロシージャ
バージョン 10.0.1 での動作の変更, 110

sa_get_request_profile システム・プロシージャ
バージョン 10.0.1 での動作の変更, 111

sa_get_request_times システム・プロシージャ
バージョン 10.0.1 での動作の変更, 111

sa_get_server_messages システム・プロシージャ
バージョン 11.0.0 で廃止予定, 71

sa_get_table_definition システム・プロシージャ
バージョン 11.0.1 の新機能, 3

sa_get_user_status システム・プロシージャ
バージョン 11.0.0 の新機能, 42

sa_index_density システム・プロシージャ
バージョン 11.0.0 の強化, 43

sa_internal_text_index_postings システム・プロシージャ
内部でのみ使用) バージョン 11.0.0 の新機能, 42

sa_load_cost_model システム・プロシージャ
バージョン 10.0.0 の新機能, 148

sa_locks システム・プロシージャ
バージョン 10.0.0 での動作の変更, 167

sa_make_object システム・プロシージャ
バージョン 10.0.0 の強化, 147

sa_materialized_view_can_be_immediate システム・プロシージャ
バージョン 11.0.0 の新機能, 43

sa_materialized_view_info システム・プロシージャ
バージョン 10.0.0 の新機能, 148
バージョン 10.0.1 での動作の変更, 110
バージョン 11.0.0 の強化, 43

sa_nchar_terms システム・プロシージャ
バージョン 11.0.0 の新機能, 42

sa_performance_diagnostics システム・プロシージャ
バージョン 10.0.0 の強化, 151

sa_post_login_procedure システム・プロシージャ
バージョン 11.0.0 の新機能, 43

sa_procedure_profile_summary システム・プロシージャ
バージョン 10.0.0 の強化, 150

sa_procedure_profile システム・プロシージャ
バージョン 10.0.0 の強化, 150

sa_refresh_materialized_views システム・プロシージャ
バージョン 10.0.0 の新機能, 148

sa_refresh_text_indexes システム・プロシージャ
バージョン 11.0.0 の新機能, 42

sa_remove_tracing_data システム・プロシージャ
バージョン 10.0.0 の新機能, 148

sa_reset_identity システム・プロシージャ
オートインクリメント・カラムのあるテーブルの再ロード, 397
バージョン 10.0.0 での動作の変更, 167

sa_save_trace_data システム・プロシージャ
バージョン 10.0.0 の新機能, 148

sa_send_udp システム・プロシージャ
バージョン 9.0.2 の新機能, 266

sa_server_messages システム・プロシージャ
バージョン 11.0.0 の新機能, 71

sa_server_option システム・プロシージャ
バージョン 10.0.0 の強化, 150
バージョン 11.0.0 の強化, 52
バージョン 9.0.1 の強化, 291

sa_set_http_option システム・プロシージャ
バージョン 10.0.0 の強化, 161

sa_set_http_option システム・プロシージャ
バージョン 11.0.1 の強化, 4

sa_set_soap_header システム・プロシージャ
バージョン 10.0.0 の新機能, 161

sa_set_tracing_level システム・プロシージャ
バージョン 10.0.0 の新機能, 148

sa_sever_option システム・プロシージャ
バージョン 11.0.0 の強化, 43

sa_snapshots システム・プロシージャ
バージョン 10.0.0 の新機能, 148

sa_split_list システム・プロシージャ
バージョン 10.0.0 の新機能, 148

sa_table_stats システム・プロシージャ
バージョン 10.0.0 の新機能, 148

sa_text_index_handles システム・プロシージャ
内部でのみ使用、バージョン 11.0.0 の新機能, 42

sa_text_index_postings システム・プロシージャ
内部でのみ使用、バージョン 11.0.0 の新機能, 42

sa_text_index_stats システム・プロシージャ
バージョン 11.0.0 の新機能, 42

sa_text_index_vocab システム・プロシージャ
バージョン 11.0.0 の新機能, 42

sa_transactions システム・プロシージャ
バージョン 10.0.0 の新機能, 130

sa_unload_cost_model システム・プロシージャ
バージョン 10.0.0 の新機能, 148

sa_validate システム・プロシージャ
バージョン 10.0.0 での動作の変更, 167
バージョン 10.0.0 で廃止予定のインデックス・オプション, 174
バージョン 10.0.0 で廃止予定のデータ・オプション, 174
バージョン 10.0.0 で廃止予定のフル・オプション, 174
バージョン 11.0.0 での動作の変更, 62
バージョン 9.0.1 の新機能, 288

sa_verify_password システム・プロシージャ
バージョン 9.0.2 の新機能, 266

SACHARSET 環境変数
バージョン 10.0.0 の新機能, 171

- SADatabase エージェント
 - バージョン 10.0.0 の新機能, 130
- SADbType 列挙体 [SA .NET 2.0]
 - バージョン 10.0.1 での動作の変更, 114
 - バージョン 10.0.1 の強化, 109
- SADIAGDIR 環境変数
 - バージョン 10.0.0 の新機能, 255
- SADIR 環境変数
 - バージョン 10.0.0 の新機能, 171
- SALANG 環境変数
 - バージョン 10.0.0 の新機能, 171
- SALOGDIR 環境変数
 - バージョン 10.0.0 の新機能, 171
- samples-dir
 - マニュアルの使用方法, xi
- SAOLEDDB
 - バージョン 10.0.0 での動作の変更, 169
- saopts.sql
 - バージョン 10.0.1 での動作の変更, 111
- SA Server エージェント
 - バージョン 10.0.0 の新機能, 130
- SA SpxOptionsBuilder クラス [SA .NET 2.0]
 - バージョン 11.0.0 でサポート終了, 69
- sasql_commit 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_connect 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_data_seek 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_disconnect 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_errorcode 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_error 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_execute 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_fetch_array 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_fetch_field 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_fetch_object 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_fetch_row 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_free_result 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_identity 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_num_fields 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_num_rows 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_pconnect 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_query 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_result_all 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_rollback 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasql_set_option 関数 (PHP)
 - バージョン 11.0.0 の新機能, 61
- sasrv.ini
 - バージョン 10.0.0 での動作の変更, 166
- SATMP 環境変数
 - バージョン 10.0.0 の新機能, 171
 - バージョン 11.0.0 の強化, 53
- sbgse2.dll
 - バージョン 10.0.1 での Ultra Light の動作の変更, 122
- scale オプション
 - バージョン 11.0.0 での動作の変更, 65
- SCHEMATA ローセット、OLE DB
 - バージョン 11.0.1 の新機能, 6
- ScoutSync
 - サポート終了, 331
- SearchBindery プロトコル・オプション
 - バージョン 11.0.0 でサポート終了, 69
- secure_feature_key オプション
 - バージョン 10.0.0 の新機能, 136
- secure_feature_key プロトコル
 - バージョン 10.0.0 の新機能, 143
- SecureFeatures プロパティ
 - バージョン 10.0.0 の新機能, 136
- Security Builder
 - バージョン 10.0.1 の強化, 124
- SELECT 文
 - バージョン 10.0.0 の強化, 153
 - バージョン 10.0.1 の強化, 105
 - バージョン 11.0.0 での動作の変更, 66
 - バージョン 11.0.0 の強化, 29, 47, 48
 - バージョン 11.0.1 の強化, 5, 6
- SELinux のポリシー

バージョン 11.0.0 の新機能, 53
SendBufferSize プロトコル・オプション
バージョン 10.0.0 の強化, 134
SendingTracingTo プロパティ
バージョン 10.0.0 の新機能, 146
SeparateCheckpointLog プロパティ
バージョン 10.0.0 で削除されたデータベース・
プロパティ, 199
SeparateForeignKeys プロパティ
バージョン 10.0.0 で削除されたデータベース・
プロパティ, 199
ServerEdition プロパティ
バージョン 11.0.1 の新機能, 3
ServerName プロパティ
バージョン 10.0.0 の新機能, 145
ServerNodeAddress プロパティ
バージョン 11.0.0 の新機能, 39
ServerPort プロトコル・オプション
バージョン 10.0.0 での動作の変更, 173
session_key
バージョン 10.0.0 の Mobile Link [mlsrv10] の新
機能, 204
SessionCreateTime プロパティ
バージョン 10.0.0 の新機能, 143
SessionID プロパティ
バージョン 10.0.0 の新機能, 143
SessionLastTime プロパティ
バージョン 10.0.0 の新機能, 143
SET_BITS 関数
バージョン 10.0.0 の新機能, 149
SET_BIT 関数
バージョン 10.0.0 の新機能, 149
SET OPTION 文
バージョン 10.0.0 での動作の変更, 251
バージョン 11.0.0 で削除, 96
SET PARTNER FAILOVER 句
バージョン 10.0.1 の新機能, 107
setup.exe
バージョン 10.0.1 での Ultra Light の動作の変
更, 122
SET 文
バージョン 10.0.0 の強化, 130
SHARED_DIR 環境変数
バージョン 11.0.0 でサポート終了, 71
Sierra Wireless Aircards
バージョン 11.0.0 でサポート終了, 77
SIMILAR TO 探索条件
バージョン 11.0.0 の新機能, 29
バージョン 11.0.1 での動作の変更, 10
SiteScriptName
サポート終了, 328
SnapshotCount プロパティ
バージョン 10.0.0 の新機能, 143, 146
SnapshotIsolationState プロパティ
バージョン 10.0.0 の新機能, 146
SNMP
バージョン 10.0.1 の強化, 109
バージョン 11.0.0 の強化, 56
SOAP_HEADER 関数
バージョン 10.0.0 の新機能, 161
SOAPHEADER 句
バージョン 10.0.0 の新機能, 161
SOAP サービス
バージョン 10.0.0 での動作の変更, 173
SORTKEY 関数
バージョン 10.0.1 の強化, 106
sp_hook_dbmlsync_all_error
バージョン 10.0.0 の新機能, 210
sp_hook_dbmlsync_communication_error
バージョン 10.0.0 の新機能, 210
sp_hook_dbmlsync_download_com_error
バージョン 10.0.0 で廃止予定, 222
sp_hook_dbmlsync_fatal_sql_error
バージョン 10.0.0 で廃止予定, 222
sp_hook_dbmlsync_log_rescan
バージョン 10.0.0 での動作の変更, 222
sp_hook_dbmlsync_misc_error
バージョン 10.0.0 の新機能, 210
sp_hook_dbmlsync_set_ml_connect_info
バージョン 10.0.1 の新機能, 116
sp_hook_dbmlsync_sql_error
バージョン 10.0.0 で廃止予定, 222
バージョン 10.0.0 の新機能, 210
Specification プロパティ
バージョン 10.0.1 の新機能, 106
SPX
バージョン 11.0.0 でサポート終了, 69
SpxOptionsBuilder プロパティ [SA .NET 2.0]
バージョン 11.0.0 でサポート終了, 69
SpxOptionsString プロパティ [SA .NET 2.0]
バージョン 11.0.0 でサポート終了, 69
SQL_BIGINT
バージョン 10.0.0 での動作の変更, 169
sql_flagger_error_level オプション

- バージョン 10.0.1 の強化, 104
 - sql_flagger_warning_level オプション
 - バージョン 10.0.1 の強化, 104
 - SQL/1992
 - バージョン 10.0.1 で廃止予定の SQL FLAGGER のサポート, 114
 - SQLANY10 環境変数
 - バージョン 11.0.0 での名前の変更, 71
 - SQLANY11 環境変数
 - バージョン 11.0.0 の新機能, 71
 - sqlanydb
 - 11.0.0 の新機能, 48
 - SQLANYSAMP10 環境変数
 - バージョン 10.0.0 の新機能, 164
 - バージョン 11.0.0 での名前の変更, 71
 - SQLANYSAMP11 環境変数
 - バージョン 11.0.0 の新機能, 71
 - SQLANYSH10 環境変数
 - バージョン 11.0.0 でサポート終了, 71
 - SQL Anywhere
 - バージョン 11.0.1 の新機能, 22
 - マニュアル, viii
 - sqlanywhere_commit 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_connect 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_data_seek 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_disconnect 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_errorcode 関数
 - バージョン 10.0.0 の新機能, 157
 - sqlanywhere_errorcode 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_error 関数
 - バージョン 10.0.0 の新機能, 157
 - sqlanywhere_error 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_execute 関数
 - バージョン 10.0.0 の新機能, 157
 - sqlanywhere_execute 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_fetch_array 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_fetch_field 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_fetch_object 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_fetch_row 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_free_result 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_identity 関数
 - バージョン 10.0.0 の新機能, 157
 - sqlanywhere_identity 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_insert_id 関数
 - バージョン 10.0.0 の新機能, 157
 - sqlanywhere_num_fields 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_num_rows 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_pconnect 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_query 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_result_all 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_rollback 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
 - sqlanywhere_set_option 関数
 - バージョン 10.0.0 の強化, 157
 - sqlanywhere_set_option 関数 (旧式)
 - バージョン 11.0.0 で廃止予定, 61
- SQL Anywhere 11
 - アップグレード, 391
- SQL Anywhere 11 CustDB データ・ソース
 - バージョン 11.0.0 での動作の変更, 100
- SQL Anywhere 11 Demo データ・ソース
 - バージョン 11.0.0 での動作の変更, 100
- SQL Anywhere Broadcast Repeater ユーティリティ [dbns10]
 - バージョン 10.0.0 の新機能, 139
 - バージョン 11.0.0 での動作の変更, 71
- SQL Anywhere C API
 - 11.0.0 の新機能, 48
- SQL Anywhere Explorer
 - バージョン 10.0.0 の新機能, 156
- SQL Anywhere for MS Access 移行ユーティリティ
 - バージョン 11.0.0 でサポート終了, 55
- SQL Anywhere MIB
 - バージョン 11.0.0 の強化, 56
- SQL Anywhere OEM 版
 - バージョン 10.0.1 での動作の変更, 111

-
- バージョン 10.0.1 でのマニュアルの変更, 113
 - SQL Anywhere PHP モジュール
 - バージョン 10.0.0 の強化, 157
 - バージョン 10.0.0 の動作の変更, 171
 - SQL Anywhere SNMP Extension Agent
 - バージョン 10.0.0 の強化, 130, 164
 - バージョン 10.0.1 の強化, 109
 - バージョン 11.0.0 の強化, 56
 - SQL Anywhere コンソール・ユーティリティ [dbconsole]
 - バージョン 10.0.0 の強化, 250
 - SQL Anywhere のアップグレード 説明, 391
 - SQL Anywhere のサポート終了機能
 - バージョン 10.0.0, 192
 - バージョン 10.0.1, 114
 - バージョン 11.0.0, 68
 - バージョン 11.0.1, 13
 - SQL Anywhere の新機能
 - バージョン 10.0.0, 129
 - バージョン 10.0.1, 103
 - バージョン 11.0.0, 29
 - SQL Anywhere の動作の変更
 - バージョン 10.0.0, 165
 - バージョン 10.0.1, 109
 - バージョン 11.0.0, 58
 - バージョン 11.0.1, 9
 - SQL Anywhere の廃止予定機能
 - バージョン 10.0.0, 192
 - バージョン 10.0.1, 114
 - バージョン 11.0.0, 68
 - バージョン 11.0.1, 13
 - SQL Anywhere パススルー・スクリプト
 - Ultra Light バージョン 11.0.0 での強化, 81
 - SQL Anywhere プラグイン
 - バージョン 10.0.0 での動作の変更, 252
 - SQL Anywhere プラグインでの照合の適合化サポート
 - バージョン 11.0.0 の新機能, 89
 - SQL API
 - バージョン 10.0.0 の QAnywhere の新機能, 225
 - SQLCA
 - バージョン 10.0.1 での Ultra Light の制限, 121
 - SQL FLAGGER
 - バージョン 10.0.1 の強化, 104
 - SQLFLAGGER 関数
 - バージョン 10.0.1 の新機能, 104
 - SQLGetConnectAttr
 - バージョン 11.0.0 での動作の変更, 62
 - SQLLOCALE 環境変数
 - バージョン 10.0.0 でサポート対象外, 193
 - SQLPATH 環境変数
 - バージョン 10.0.0 での動作の変更, 174
 - sqlpp ユーティリティ
 - バージョン 10.0.1 の強化, 104
 - SQL Remote
 - ASE サポートの削除, 231
 - ASE サポートの廃止, 280
 - Mobile Link 移行, 231
 - アップグレード, 429
 - SQL Remote for ASE
 - バージョン 10.0.0 での削除, 231
 - SQL Remote の新機能
 - バージョン 10.0.0, 231
 - バージョン 11.0.0, 80
 - バージョン 9.0.2, 270
 - SQL Remote の動作の変更
 - バージョン 10.0.0, 231
 - バージョン 10.0.1, 120
 - バージョン 11.0.0, 80
 - バージョン 9.0.2, 280
 - SQL SECURITY 句
 - バージョン 11.0.0 の新機能, 42
 - SQL サポート
 - バージョン 11.0.1 での Ultra Light J の強化, 20
 - SQL パススルー
 - バージョン 11.0.0 の新機能, 74
 - SQL プリプロセッサ
 - バージョン 10.0.1 の強化, 104
 - SQL 文の実行
 - Interactive SQL バージョン 11.0.0 の強化, 93
 - SSL
 - バージョン 11.0.0 での動作の変更, 63
 - ssqueue
 - 9.0.2 で廃止予定, 280
 - バージョン 10.0.0 での削除, 231
 - ssremote
 - 9.0.2 で廃止予定, 280
 - バージョン 10.0.0 での削除, 231
 - ssxtract
 - バージョン 10.0.0 での削除, 231
 - START AT 句
 - バージョン 10.0.0 での Ultra Light の強化, 238
 - START DATABASE 文
-

- バージョン 10.0.0 の強化, 154
- バージョン 10.0.1 の強化, 105
- StartDBPermission プロパティ
 - バージョン 10.0.0 の新機能, 145
- START SYNCHRONIZATION DELETE 文
 - バージョン 10.0.0 での Ultra Light の強化, 238
- StatementDescribes プロパティ
 - 接続プロパティ、11.0.1 の新機能, 2
 - データベース・プロパティ、11.0.1 の新機能, 2
- StatementPostAnnotatesSimple プロパティ
 - 接続プロパティ、11.0.1 の新機能, 2
 - データベース・プロパティ、11.0.1 の新機能, 2
- StatementPostAnnotatesSkipped プロパティ
 - 接続プロパティ、11.0.1 の新機能, 2
 - データベース・プロパティ、11.0.1 の新機能, 2
- StatementPostAnnotates プロパティ
 - 接続プロパティ、11.0.1 の新機能, 2
 - データベース・プロパティ、11.0.1 の新機能, 2
- STOP SYNCHRONIZATION DELETE 文
 - バージョン 10.0.0 での Ultra Light の強化, 238
- StreamErrorParameters プロパティ
 - バージョン 11.0.1 での Ultra Light の強化, 18
- StreamsUsed プロパティ
 - バージョン 11.0.0 の新機能, 40
- string_rtruncation オプション
 - バージョン 10.0.0 での動作の変更, 168, 175
- StringHistogramsFix プロパティ
 - バージョン 10.0.0 で削除されたデータベース・プロパティ, 199
- SUBSTR 関数
 - バージョン 10.0.0 の QAnywhere の新機能, 228
- Sybase Central
 - データベースのアップグレード, 405
 - バージョン 10.0.0 の強化, 248
 - バージョン 9.0.1 の新機能, 292
 - 古いデータベースの管理, 392
- Sybase Central の新機能
 - バージョン 10.0.0, 248
 - バージョン 10.0.1 での SQL Anywhere プラグインの強化, 107
 - バージョン 11.0.0, 88
 - バージョン 11.0.1, 22
- Sybase Central の動作の変更
 - バージョン 10.0.0, 250
 - バージョン 11.0.0, 91
 - バージョン 11.0.1, 24
- Sybase Central のモデル・モード
 - バージョン 10.0.0 の Mobile Link の新機能, 201
 - バージョン 10.0.1 の Mobile Link の新機能, 115
- Sybase リレー・サーバのホスティング・サービス
 - バージョン 11.0.0 の新機能, 73
- SYNC_PROFILE_OPTION_VALUE 関数 [システム]
 - バージョン 11.0.1 での Ultra Light の強化, 18
- syncase125.sql
 - バージョン 10.0.0 で削除, 202
- synchronize_mirror_on_commit オプション
 - バージョン 10.0.0 の新機能, 129, 140
- synchronize_mirror_on_commit プロパティ
 - バージョン 10.0.0 の新機能, 143
- syncsa.sql
 - バージョン 10.0.0 の Mobile Link の新しい動作, 202
- SYNC ゲートウェイ
 - バージョン 10.0.0 の新しいサーバ起動同期機能, 212
- SYSATTRIBUTE
 - バージョン 10.0.0 で削除されたシステム・テーブル, 189
- SYSATTRIBUTENAME
 - バージョン 10.0.0 で削除されたシステム・テーブル, 189
- SYSCOLLATION
 - バージョン 10.0.0 で廃止予定のシステム・テーブル, 187
- SYSCOLLATIONMAPPINGS
 - バージョン 10.0.0 で廃止予定のシステム・テーブル, 187
- SYSCOLUMN
 - 互換ビュー、バージョン 10.0.0 での動作の変更, 190
 - バージョン 10.0.0 で廃止予定のシステム・テーブル, 187
- SYSCOLUMNS
 - 統合ビュー、バージョン 10.0.0 での動作の変更, 190
- SYSCONSTRAINT
 - バージョン 10.0.0 でのシステム・ビューの動作の変更, 190
- SYSDEPENDENCY

バージョン 10.0.0 の新しいシステム・ビュー, 185	バージョン 10.0.0 でのシステム・ビューの動作 の変更, 190
SYSSEXTENT バージョン 10.0.0 で削除されたシステム・テ ーブル, 189	SYSJARCOMPONENT バージョン 10.0.0 でのシステム・ビューの動作 の変更, 190
SYSSEXTERNLOGINS バージョン 10.0.0 で名前が変更されたシステ ム・テーブル, 189	SYSJAVACLASS システム・ビュー、バージョン 10.0.0 での動作 の変更, 170 バージョン 10.0.0 でのシステム・ビューの動作 の変更, 190
SYSFILE バージョン 10.0.0 でのシステム・ビューの動作 の変更, 190	SYSLOGIN バージョン 10.0.0 で削除されたシステム・テ ーブル, 189
SYSFILE システム・ビュー バージョン 11.0.0 での動作の変更, 68	SYSLOGINMAP バージョン 10.0.0 でのシステム・ビューの動作 の変更, 190 バージョン 10.0.0 の新しいシステム・ビュー, 185 バージョン 11.0.0 の新機能, 58
SYSFKCOL バージョン 10.0.0 で廃止予定のシステム・テ ーブル, 187	SYSLOGINPOLICY バージョン 11.0.0 の新機能, 58
SYSFKEY バージョン 10.0.0 の新しいシステム・ビュー, 185	SYSLOGINPOLICYOPTION バージョン 11.0.0 の新機能, 58
SYSFOREIGNKEY バージョン 10.0.0 で廃止予定のシステム・テ ーブル, 187	SYSMVOPTION バージョン 10.0.0 の新しいシステム・ビュー, 185
SYSHISTORY バージョン 10.0.0 のシステム・ビューの強化, 135 バージョン 9.0.1 の新機能のシステム・テ ーブル, 291	SYSMVOPTIONNAME バージョン 10.0.0 の新しいシステム・ビュー, 185
SYSIDX バージョン 10.0.0 の新しいシステム・ビュー, 185	sysncsa.sql バージョン 10.0.0 での sysncsa.sql への名前の変 更, 224
SYSIDXCOL バージョン 10.0.0 の新しいシステム・ビュー, 185	SYSOBJECT バージョン 10.0.0 の新しいシステム・ビュー, 185 バージョン 11.0.0 での動作の変更, 58
SYSINDEX バージョン 10.0.0 でのシステム・ビューの動作 の変更, 198 バージョン 10.0.0 で廃止予定のシステム・テ ーブル, 187	SYSOPTBLOCK バージョン 10.0.0 で削除されたシステム・テ ーブル, 189
SYSINDEXES バージョン 11.0.0 での動作の変更, 58	SYSOPTJOINSTRATEGIES バージョン 10.0.0 で削除されたシステム・ ビュー, 189
SYSINFO 互換ビュー、バージョン 10.0.0 での動作の変 更, 170	SYSOPTJOINSTRATEGY バージョン 10.0.0 で削除されたシステム・テ ーブル, 189
SYSIXCOL バージョン 10.0.0 で廃止予定のシステム・テ ーブル, 187	SYSOPTORDER
SYSJAR	

- バージョン 10.0.0 で削除されたシステム・テーブル, 189
- SYSOPTORDERS**
バージョン 10.0.0 で削除されたシステム・ビュー, 189
- SYSOPTQUANTIFIER**
バージョン 10.0.0 で削除されたシステム・テーブル, 189
- SYSOPTREQUEST**
バージョン 10.0.0 で削除されたシステム・テーブル, 189
- SYSOPTREWRITE**
バージョン 10.0.0 で削除されたシステム・テーブル, 189
- SYSPHYSIDX**
バージョン 10.0.0 の新しいシステム・ビュー, 185
- SYSPROCEDURES**
バージョン 10.0.0 で削除されたビュー, 189
- SYSPROCARM**
バージョン 10.0.0 でのシステム・ビューの動作の変更, 190
- SYSPROCARMS**
バージョン 10.0.0 での統合ビューの動作の変更, 191
- SYSPROCS**
バージョン 10.0.0 の新しいシステム・ビュー, 185
- SYSPROXYTAB**
バージョン 10.0.0 の新しいシステム・ビュー, 185
- SYSPUBLICATION**
バージョン 10.0.0 でのシステム・ビューの動作の変更, 210
- SYSREMOTEOPTION**
バージョン 10.0.0 でのシステム・ビューの動作の変更, 190
- SYSREMOTUSER**
バージョン 10.0.0 でのシステム・ビューの動作の変更, 191
- SYSSEVERERS**
バージョン 10.0.0 で名前が変更されたシステム・テーブル, 189
- SYSOURCE**
バージョン 10.0.0 の新しいシステム・ビュー, 185
- SYSUBSCRIPTION**
バージョン 10.0.0 でのシステム・ビューの動作の変更, 191
- SYSSYNC**
バージョン 10.0.0 でのシステム・ビューの動作の変更, 191
- SYSSYNCSRIPT**
バージョン 10.0.0 の新しいシステム・ビュー, 185
- SYSTAB**
バージョン 10.0.0 のシステム・ビューの強化, 164
- SYSTABCOL**
バージョン 10.0.0 の新しいシステム・ビュー, 185
- SYSTABLE**
バージョン 10.0.0 で廃止予定, 193
バージョン 10.0.0 で廃止予定のシステム・テーブル, 187
- SYSTEXTCONFIG**
バージョン 11.0.0 の新機能, 58
- SYSTEXTIDX**
バージョン 11.0.0 の新機能, 58
- SYSTEXTIDXTAB**
バージョン 11.0.0 の新機能, 58
- SYSUSER**
バージョン 10.0.0 の新しいシステム・ビュー, 185
- SYSUSERAUTH**
バージョン 10.0.0 で廃止予定のシステム・テーブル, 187
- SYSUSERAUTHORITY**
バージョン 10.0.0 の新しいシステム・ビュー, 185
- SYSUSERLIST**
バージョン 10.0.0 で廃止予定のシステム・テーブル, 187
- SYSUSERMESSAGES**
バージョン 10.0.0 で名前が変更されたシステム・テーブル, 189
- SYSUSERPERM**
バージョン 10.0.0 で廃止予定のシステム・テーブル, 187
- SYSUSERPERMS**
バージョン 10.0.0 で廃止予定のシステム・テーブル, 187

T

TableBitMaps プロパティ
バージョン 10.0.0 で削除されたデータベース・
プロパティ, 199

TableOrderChecking dbmlsync 拡張オプション
バージョン 10.0.0 の新機能, 211

TablesQualTriggers プロパティ
バージョン 10.0.0 で削除されたデータベース・
プロパティ, 199

TCP/IP
バージョン 10.0.0 での動作の変更, 172, 173

TcpIpAddresses プロパティ
バージョン 11.0.0 の新機能, 40

TDS DATE
バージョン 10.0.1 の新機能, 108

TDS TIME
バージョン 10.0.1 の新機能, 108

temp_space_limit_check オプション
バージョン 10.0.0 での動作の変更, 175

timeout プロトコル・オプション
10.0.0 での Mobile Link クライアントの新しい
プロトコル・オプション, 209

timestamp_format オプション
バージョン 10.0.0 での動作の変更, 175

TLS
バージョン 10.0.0 の Ultra Light の新しい暗号化
タイプ, 235
バージョン 11.0.0 での動作の変更, 63

TOP 句
バージョン 9.0.1 の変数の強化サポート, 287

TRACED_PLAN 関数
バージョン 10.0.0 の新機能, 149

TransactionsSpanLogs プロパティ
バージョン 10.0.0 で削除されたデータベース・
プロパティ, 199

TransactionStartTime プロパティ
バージョン 9.0.1 の新機能, 293

Transact-SQL 外部ジョイン
バージョン 11 へのアップグレードのトラブル
シューティング, 404

Treo 650
バージョン 10.0.0 で追加されたサポート, 214

truncate_date_values option オプション
バージョン 9.0.2 で廃止予定, 277

truncate_date_values オプション
バージョン 10.0.0 での動作の変更, 176

truncate_with_auto_commit オプション
バージョン 11.0.0 でサポート終了, 69

TRUNCATE TEXT INDEX 文
バージョン 11.0.0 の新機能, 46

tsql_hex_constant オプション
バージョン 11.0.0 でサポート終了, 69

tsql_outer_joins プロパティ
バージョン 10.0.0 の新機能, 143

U

UL_STORE_PARMS マクロ
9.0.0 での Ultra Light の動作の変更, 332

ul_stream_error 構造体
Ultra Light バージョン 11.0.0 での強化, 83

ul_synch_info
Ultra Light バージョン 11.0.0 での動作の変更,
87

ulafreg ユーティリティ
バージョン 10.0.1 での動作の変更, 122

ulcond.log
バージョン 10.0.0 での Ultra Light の動作の変
更, 247

ulcond10 ユーティリティ
バージョン 10.0.0 での Ultra Light の強化, 237

ulconv ユーティリティ
9.0.1 の Ultra Light の新機能, 298
バージョン 10.0.0 での削除, 245

ULEnableGenericSchema 関数
9.0.1 の廃止予定機能, 304

ulgen ユーティリティ
Ultra Light データベースのアップグレード,
425
バージョン 10.0.0 での削除, 245

ulinfo ユーティリティ
バージョン 10.0.0 の Ultra Light の新機能, 237

ulinit ユーティリティ
データベースのアップグレード手順, 427
バージョン 10.0.0 での Ultra Light の強化, 237

ulsql ユーティリティ
バージョン 10.0.0 での削除, 245

ULjDbT
Ultra Light J ユーティリティ, 20

ulload ユーティリティ
データベースのアップグレード手順, 423
バージョン 10.0.0 での Ultra Light の強化, 237

ulmbvreg

- バージョン 10.0.0 での ulafreg への変更, 247
- ulodbc サーバ・クラス
 - バージョン 11.0.0 の新機能, 55
- ULPalmExit 関数
 - 9.0.1 の廃止予定機能, 303
- ULPalmLaunch 関数
 - 9.0.1 の廃止予定機能, 303
- ULRegisterErrorCallback 関数 [UL C/C++]
 - 9.0.1 の Ultra Light C/C++ の新機能, 298
- ULSQLCONNECT 環境変数
 - バージョン 10.0.0 の Ultra Light の新機能, 238
- ulsync ユーティリティ
 - バージョン 10.0.0 での Ultra Light の強化, 238
- Ultra Light
 - データベースとアプリケーションのアップグレード, 419
- Ultra Light.NET
 - バージョン 10.0.0 での Ultra Light の強化, 243
- Ultra Light .NET
 - Ultra Light バージョン 11.0.0 での強化, 82, 86
- Ultra Light.NET コントロール
 - 9.0.1 の新機能, 297
- Ultra Light ALTER DATABASE SCHEMA FROM FILE 文
 - Ultra Light バージョン 11.0.0 の新機能, 83
- Ultra Light ALTER SYNCHRONIZATION PROFILE 文
 - Ultra Light、バージョン 11.0.0 の新機能, 81
- Ultra Light C/C++ API
 - Ultra Light バージョン 11.0.0 での強化, 85
- Ultra Light CREATE SYNCHRONIZATION PROFILE 文
 - Ultra Light バージョン 11.0.0 の新機能, 81
- Ultra Light DROP SYNCHRONIZATION PROFILE 文
 - Ultra Light バージョン 11.0.0 の新機能, 81
- Ultra Light Embedded SQL
 - 9.0.2 で廃止予定, 281
 - バージョン 10.0.0 の新機能, 234
- Ultra Light ESQL
 - Ultra Light バージョン 11.0.0 での強化, 86
- Ultra Light for C/C++
 - バージョン 10.0.0 での Ultra Light の強化, 241
- Ultra Light for M-Business Anywhere
 - 9.0.1 の新機能, 297
 - Ultra Light バージョン 11.0.0 での強化, 86
- バージョン 10.0.0 での Ultra Light の強化, 244
- Ultra Light isolation_level オプション
 - Ultra Light バージョン 11.0.0 の新機能, 82
- Ultra Light J
 - ULjDbT ユーティリティ, 20
 - Ultra Light J データベース転送ユーティリティ, 20
 - Ultra Light バージョン 11.0.0 での強化, 86
- Ultra Light J データベース転送ユーティリティ BlackBerry, 20
- Ultra Light J の新機能
 - バージョン 11.0.1, 20
- Ultra Light J の動作の変更
 - バージョン 11.0.1, 19, 21
- Ultra Light LOAD TABLE 文
 - Ultra Light バージョン 11.0.0 の新機能, 83
- Ultra Light SELECT 文
 - Ultra Light バージョン 11.0.0 の新機能, 81
- Ultra Light SQL
 - バージョン 10.0.0 での Ultra Light の強化, 242
 - マルチテーブル・ジョイン, 323
- Ultra Light SYNCHRONIZE 文
 - Ultra Light バージョン 11.0.0 の新機能, 81
- Ultra Light XML のデータベース・アンロード・ユーティリティ
 - バージョン 10.0.0 での Ultra Light の強化, 238
- Ultra Light XML のデータベース・ロード・ユーティリティ
 - バージョン 10.0.0 での Ultra Light の強化, 237
- Ultra Light アプリケーション
 - Ultra Light バージョン 11.0.0 の新機能、ロー・トラバーサル, 85
 - アップグレード・パス, 425
- Ultra Light エンジン
 - 9.0.1 の Ultra Light の新機能, 298
- Ultra Light クライアント
 - Ultra Light バージョン 11.0.0 での強化, 83
- Ultra Light コンポーネント
 - 9.0.1 でのテーブル API の動作の変更, 304
- Ultra Light 情報ユーティリティ
 - バージョン 10.0.0 の Ultra Light の新機能, 237
- Ultra Light 静的型 C++ API
 - 9.0.2 で廃止予定, 281
- Ultra Light 静的型 Java API
 - 9.0.2 で廃止予定, 281
- Ultra Light データベース
 - アップグレード・パス, 421

-
- 接続方法, 420
 - バージョン 10.0.0 の新機能, 233
 - バージョン 10.0.1 での Ultra Light 接続の強化, 121
 - バージョン 11.0.0 の強化, 81
 - バージョン 11.0.0 の新機能, 81
 - Ultra Light データベース検証ユーティリティ (ulvalid)
 - Ultra Light バージョン 11.0.0 の新機能, 82
 - Ultra Light データベース・コンバータ
 - 9.0.1 の Ultra Light の新機能, 298
 - Ultra Light データベース作成ユーティリティ
 - バージョン 10.0.0 での Ultra Light の強化, 237
 - Ultra Light データベース初期化ユーティリティ
 - バージョン 10.0.0 での Ultra Light の強化, 237
 - Ultra Light データベース初期化ユーティリティ (ulinit)
 - Ultra Light バージョン 11.0.0 での強化, 84
 - 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 バージョン 11.0.0 での強化, 84
 - Ultra Light テーブル
 - Ultra Light バージョン 11.0.0 の新機能、ロー・トラバーサル, 85
 - Ultra Light テーブル API
 - Ultra Light バージョン 11.0.0 での強化, 85
 - Ultra Light 同期ユーティリティ
 - バージョン 10.0.0 での Ultra Light の強化, 238
 - Ultra Light の新機能
 - バージョン 10.0.0, 233
 - バージョン 10.0.1, 121
 - バージョン 11.0.0, 81
 - バージョン 11.0.1, 18
 - バージョン 9.0.2, 270
 - Ultra Light のデータベース
 - ファイル・フォーマットのアップグレード, 420
 - Ultra Light の動作の変更
 - バージョン 10.0.0, 244
 - バージョン 10.0.1, 122
 - バージョン 11.0.0, 86
 - バージョン 9.0.2, 280
 - Ultra Light プラグイン
 - バージョン 10.0.0 の新機能, 249
 - [Ultra Light プラン] タブ
 - バージョン 10.0.0 で削除, 252
 - Ultra Light 古いデータベースのアンロード・ユーティリティ
 - バージョン 10.0.0 の Ultra Light の新機能, 237
 - UltraLite_Connection オブジェクト
 - Ultra Light バージョン 11.0.0 の新機能、ロー・トラバーサル, 85
 - ulunloadold ユーティリティ
 - データベースのアップグレード手順, 423
 - バージョン 10.0.0 の Ultra Light の新機能, 237
 - ulunload ユーティリティ
 - バージョン 10.0.0 での Ultra Light の強化, 238
 - ULUtil
 - バージョン 10.0.0 での ULDBUtil への名前の変更, 247
 - ulvalid
 - Ultra Light バージョン 11.0.0 の新機能, 82
 - ulview ユーティリティ
 - バージョン 10.0.0 での削除, 245
 - ulxml ユーティリティ
 - バージョン 10.0.0 での削除, 245
 - Unicode
 - Ultra Light アップグレードの推奨事項, 420
 - UNION 演算子
 - バージョン 10.0.0 での Ultra Light の強化, 238
 - UNION 文
 - バージョン 10.0.1 の強化, 105
 - バージョン 11.0.0 での動作の変更, 66
 - バージョン 11.0.0 の強化, 48
 - UniqueClientAddresses プロパティ
 - バージョン 10.0.0 の新機能, 168
 - UNIQUEIDENTIFIER データ型
 - バージョン 10.0.0 の強化, 156
 - UniqueIdentifier プロパティ
 - バージョン 10.0.0 で削除されたデータベース・プロパティ, 199
 - バージョン 11.0.0 でサポート終了, 68
 - UNIX
 - バージョン 10.0.0 での動作の変更, 172
 - バージョン 10.0.0 の新機能, 159
 - バージョン 10.0.1 の強化, 108
 - バージョン 11.0.0 の新機能, 53
 - unknown_timeout オプション
 - バージョン 10.0.0 で置き換えられた Mobile Link クライアント・プロトコル・オプション, 209
 - UNLOAD TABLE 文
 - バージョン 10.0.0 の強化, 153
-

バージョン 11.0.0 の強化, 45, 47
バージョン 9.0.1 の変数の強化サポート, 287
UNLOAD 文
 バージョン 11.0.0 の強化, 47
 バージョン 9.0.1 の変数の強化サポート, 287
UpdatedRowSource プロパティ [SA .NET 2.0]
 バージョン 10.0.1 での動作の変更, 111
UPDATE 文
 バージョン 10.0.1 の強化, 105, 108
 バージョン 11.0.0 での動作の変更, 66
 バージョン 11.0.0 の強化, 48, 55
 バージョン 11.0.1 の強化, 6
UPDLOCK テーブル・ヒント
 バージョン 10.0.0 の新機能, 153
upload_cursor
 バージョン 10.0.0 で削除, 214
 バージョン 9.0.0 で廃止予定, 329
upload_deleted_rows
 バージョン 10.0.0 の新しい動作, 218
upload_inserted_rows
 バージョン 10.0.0 の新しい動作, 218
upload_updated_rows
 バージョン 10.0.0 の新しい動作, 218
user_estimates オプション
 バージョン 11.0.0 の強化, 48
UserAppInfo プロパティ
 バージョン 9.0.1 の新機能, 293
UTF-16 エンコード
 CSCONVERT 関数, 54
 LOAD TABLE 文, 54
 UNLOAD 文, 54
 バージョン 11.0.0 の強化, 54
UTF-8
 バージョン 10.0.0 での Ultra Light の強化, 235
UTF-8
 Ultra Light アップグレードの推奨事項, 420
 URL のサポート、バージョン 11.0.0 での動作の変更, 49
UTF8BIN 照合
 バージョン 10.0.0 の新機能, 163
UTF8 照合
 バージョン 10.0.0 で廃止予定, 193
util_db.ini
 バージョン 10.0.0 で廃止予定, 199
UUID
 8.0.2 での Mobile Link の新機能, 339
uuid_has_hyphens オプション

バージョン 10.0.0 の新機能, 140
バージョン 11.0.0 でサポート終了, 69

V

V4T モード
 9.0.1 の Ultra Light の新機能, 298
VALIDATE CHECKSUM 文
 バージョン 9.0.1 の新機能, 288
VALIDATE DATABASE 文
 バージョン 10.0.0 の新機能, 152
VALIDATE INDEX 文
 バージョン 10.0.0 の強化, 154
VALIDATE MATERIALIZED VIEW 文
 バージョン 10.0.0 の新機能, 152
VALIDATE TABLE 文
 バージョン 10.0.0 で廃止予定のオプション, 192
VALIDATE 権限
 バージョン 10.0.0 の新機能, 136
VALIDATE 文
 バージョン 10.0.0 での動作の変更, 192
 バージョン 10.0.0 の強化, 152
ValuePtr パラメータ
 バージョン 10.0.0 の強化, 130
VARBIT データ型
 バージョン 10.0.0 の新機能, 155
VariableHashSize プロパティ
 バージョン 10.0.0 で削除されたデータベース・プロパティ, 199
varray (Oracle)
 バージョン 11.0.1 の新機能, 14
verify_password_function オプション
 バージョン 10.0.0 の新機能, 140
verify_password_function プロパティ
 バージョン 10.0.0 の新機能, 143
Veritas Cluster Server エージェント
 バージョン 10.0.0 の新機能, 130
VersionStorePages プロパティ
 バージョン 10.0.0 の新機能, 146
VFS
 バージョン 10.0.0 での Ultra Light の強化, 237
viewcert ユーティリティ
 バージョン 10.0.1 の新機能, 123
VIM メッセージ・タイプ
 SQL Remote バージョン 10.0.1 でサポート廃止予定, 120
 バージョン 11.0.0 でサポート終了, 80

Vista

SQL Anywhere 10.0.0 実行の既知の問題, 256

バージョン 10.0.1 でサポート, 124

バージョン 11.0.0 の強化, 33

Visual Studio

バージョン 11.0.0 での動作の変更, 100

Visual Studio .NET

バージョン 11.0.0 での動作の変更, 100

VSS

バージョン 11.0.0 の新機能, 33

VxWorks

サポート終了, 331

W

WaitStartTime プロパティ

接続プロパティ、11.0.1 の新機能, 2

WaitType

接続プロパティ、11.0.1 の新機能, 2

WebClientLogFile プロパティ

バージョン 11.0.0 の新機能, 52

WebClientLogging プロパティ

バージョン 11.0.0 の新機能, 52

Web サーバ

バージョン 10.0.0 の強化, 160

バージョン 11.0.0 での動作の変更, 63

Web サービス

バージョン 10.0.0 の QAnywhere の新機能, 225

バージョン 10.0.0 の強化, 160

バージョン 10.0.1 の強化, 106

バージョン 11.0.0 での動作の変更, 63

win32 ディレクトリ

バージョン 11.0.0 での動作の変更, 100

Windows

Ultra Light アップグレードの考慮事項, 420

バージョン 10.0.0 での動作の変更, 172

バージョン 11.0.0 での動作の変更, 66

Windows 95

バージョン 10.0.0 でサポート対象外, 257

Windows 98

バージョン 10.0.0 でサポート対象外, 257

Windows CE から Windows Mobile への変更

バージョン 11.0.0 での動作の変更, 99

Windows Me

バージョン 10.0.0 でサポート対象外, 257

Windows Mobile

Certicom Security Builder GSE のバージョン、

バージョン 10.0.1 の強化, 124

Ultra Light アップグレードの考慮事項, 420

Ultra Light の直接サポート, 234

バージョン 10.0.0 の強化, 158

バージョン 10.0.1 での Ultra Light FIPS サポートの変更, 122

バージョン 11.0.0 の強化, 53, 99

Windows Mobile の予約スタック・サイズ

バージョン 11.0.0 での動作の変更, 79

Windows NT

バージョン 10.0.0 でサポート対象外, 257

Windows Vista

SQL Anywhere 10.0.0 実行の既知の問題, 256

バージョン 11.0.0 の強化, 33

Windows パフォーマンス・モニタ

バージョン 10.0.0 で Mobile Link のサポート終了, 224

バージョン 11.0.0 の強化, 38

Winsock

バージョン 10.0.0 の強化, 199

WITH (XLOCK) 機能

バージョン 9.0.1 の強化, 289

WITH CONTENT LOGGING 句

バージョン 11.0.0 の新機能, 30, 44

WITH FILE NAME LOGGING 句

バージョン 11.0.0 の新機能, 45

WITH HASH SIZE 句

バージョン 10.0.0 で削除, 193

WITH ROW LOGGING 句

バージョン 11.0.0 の新機能, 45

WITH SHARE MODE 句、REFRESH

MATERIALIZED VIEW 文

バージョン 11.0.0 の新機能, 32

WRITE_CLIENT_FILE 関数

バージョン 11.0.0 の新機能, 41

WRITECLIENTFILE 権限

バージョン 11.0.0 の新機能, 36, 41

X

X64 ディレクトリ

バージョン 11.0.0 での動作の変更, 100

xp_cmdshell システム・プロシージャ

10.0.0 での動作変更, 167

xp_read_file システム・プロシージャ

バージョン 10.0.0 での動作の変更, 167

xp_scanf システム・プロシージャ

バージョン 10.0.0 での動作の変更, 167

xp_sendmail システム・プロシージャ

- バージョン 10.0.0 の強化, 151
- xp_sprintf システム・プロシージャ
 - バージョン 10.0.0 での動作の変更, 167
- xp_startsmtp システム・プロシージャ
 - バージョン 10.0.0 の強化, 151
- xp_write_file システム・プロシージャ
 - バージョン 10.0.0 での動作の変更, 167
- XPathCompiles プロパティ
 - バージョン 10.0.0 の新機能, 146
- X-Window Server
 - バージョン 10.0.0 での動作の変更, 173

あ

- アイコン
 - ヘルプでの使用, xiii
- アクセシビリティ
 - バージョン 8.0.0 の新機能, 364
- アクセント記号の区別
 - バージョン 10.0.1 での動作の変更, 113
- アスタリスク
 - 11.0.1 での全文検索の動作の変更, 9
- 圧縮されたカラム
 - バージョン 10.0.0 の新機能, 133
 - バージョン 11.0.0 の強化, 54
- 圧縮データベース
 - バージョン 10.0.0 でサポート対象外, 194
- アップグレード
 - Mobile Link, 410
 - Mobile Link システム・テーブル, 411
 - Mobile Link 統合データベース, 411
 - Mobile Link サーバ, 415
 - QAnywhere, 418
 - SQL Anywhere, 392
 - SQL Remote, 429
 - Ultra Light アプリケーション・コード, 424
 - Ultra Light アプリケーション・パス, 425
 - Ultra Light データベースのパス, 421
 - Ultra Light のデータベースとアプリケーションの概要, 419
 - Ultra Light のデータベース・ファイル・フォーマット, 420
 - 注意事項, 395
 - データベース・ミラーリング, 407
 - 同期スクリプト, 417
 - バージョン 10.0.0 での動作の変更, 165
 - バージョン 11 のデータベースの再構築, 399, 405

- バージョン 11 へのアップグレードについて, 391
- マテリアライズド・ビューが含まれるデータベース, 392
- ミラーリング・システム内のデータベース, 407
- リモート・データベース, 415
- アップグレード・ユーティリティ [dbupgrad]
 - バージョン 10.0.0 での動作の変更, 165, 170
 - バージョン 11.0.1 の強化, 3
- アプリケーション
 - Ultra Light アップグレード, 424
- アプリケーションのアップグレード
 - Ultra Light アップグレード・パス, 425
 - Ultra Light でのプロセス, 424
 - 接続コード, 424
- アプリケーション・プロファイリング
 - バージョン 10.0.0 の新機能, 130
- 暗号化
 - バージョン 10.0.1 での動作の変更, 109
 - バージョン 10.0.1 の強化, 103
 - バージョン 11.0.0 の強化, 34, 54
- 暗号化キー
 - バージョン 10.0.0 の強化, 153
- 暗号化接続パラメータ
 - バージョン 10.0.0 での動作の変更, 176
- アンロード
 - バージョン 10.0.0 の強化, 139
 - 変数へ, バージョン 11.0.0 の新機能, 31
- アンロード・ユーティリティ [dbunload]
 - データベース・ファイルのアップグレード, 402
 - バージョン 10.0.0 での動作の変更, 170, 171
 - バージョン 10.0.0 の強化, 139
 - バージョン 10.0.1 での動作の変更, 113
 - バージョン 11 へのアップグレードのトラブルシューティング, 404
 - バージョン 11.0.0 での動作の変更, 66
 - バージョン 11.0.0 の強化, 36
 - バージョン 11.0.1 の強化, 3
 - バージョン 9.0.1 の強化, 290
 - バージョン 9.0.1 の廃止予定機能, 301
- アーカイブのバックアップ
 - バージョン 11.0.1 の強化, 7
 - バージョン 9.0.1 の強化, 290
- アーカイブ・メッセージ・ストア
 - バージョン 11.0.0 の強化, 79

い

移行

SQL Remote から Mobile Link への移行, 231

移植

Ultra Light アプリケーションのアップグレード, 424

意図的ロック

バージョン 10.0.0 の新機能, 132

イベント

バージョン 11.0.1 の強化, 22

イベント・フック

9.0.1 の Mobile Link の新しい動作, 303

イベント・ログ

バージョン 10.0.0 の強化, 154

イメージのバックアップ

バージョン 9.0.1 の強化, 290

インクリメンタル・アップロード／ダウンロード

バージョン 11.0.0 の強化, 78

印刷

Interactive SQL バージョン 11.0.0 の強化, 93

インストール・ディレクトリ

バージョン 11.0.0 での動作の変更, 100

インタフェース

バージョン 10.0.0 の Ultra Light の新機能, 236

インデックス

バージョン 10.0.0 での動作の変更, 168

バージョン 10.0.0 の強化, 162

バージョン 11.0.0 での動作の変更, 67

バージョン 11.0.0 でのパフォーマンス向上, 54

バージョン 11.0.0 の強化, 30

インデックス共有処理

バージョン 10.0.0 の新機能, 162

インデックス・コンサルタント

バージョン 10.0.0 の強化, 162

インデックス・ヒント

バージョン 9.0.1 の新機能, 289

インポート・ウィザード

バージョン 11.0.0 の強化, 95

バージョン 9.0.1 の強化, 292

イン・メモリ・モード

バージョン 11.0.0 の新機能, 38

引用符付き識別子

バージョン 11.0.0 での動作の変更, 65

う

ウィザード

バージョン 10.0.0 の Ultra Light の新機能, 236

バージョン 11.0.0 の新しいウィザード, 89

バージョン 11.0.0 の強化, 92

ウィンドウ機能

バージョン 9.0.1 の新機能, 286

ウォッチ・リスト

バージョン 11.0.0 の新機能, 43

え

エクスポート・ウィザード

バージョン 11.0.0 の強化, 95

バージョン 9.0.1 の強化, 292

エラー処理

9.0.1 の Ultra Light C/C++ の強化, 298

エラー文字列

バージョン 10.0.0 での Ultra Light での強化, 237

エラー・レポート

バージョン 11.0.0 の強化, 99

バージョン 10.0.0 の新機能, 255

バージョン 10.0.1 の強化, 109

演算子の優先度、全文検索

11.0.1 での動作の変更, 9

エンディアン

11.0.1 での動作の変更, 12

バージョン 11.0.0 の強化, 54

エンドツーエンド暗号化

Ultra Light バージョン 11.0.0 での強化, 84

バージョン 11.0.0 の新機能, 75

お

大文字と小文字の区別

Ultra Light のアップグレードの考慮事項, 420

バージョン 10.0.1 での動作の変更, 113

お気に入りリスト

バージョン 11.0.0 の新機能, 90, 91

オプションのウォッチ・リスト

バージョン 11.0.0 の新機能, 43

オブティマイザ

バージョン 11.0.0 の強化, 30

バージョン 11.0.1 の強化, 6

オンライン・マニュアル

PDF, viii

オートコミット

バージョン 10.0.1 での Ultra Light の強化, 122

バージョン 11.0.0 での動作の変更, 68

か

外部アンロード
バージョン 11.0.0 の強化, 56

外部キー制約
バージョン 10.0.0 での動作の変更, 168

[外部キーのプロパティ] ウィンドウ
バージョン 9.0.1 の強化, 292

外部ジョイン
バージョン 10.0.0 での動作の変更, 193
バージョン 11 へのアップグレードのトラブル
シューティング, 404

[概要] ウィンドウ枠
バージョン 11.0.0 の新機能, 89

カスタム照合
内部アンロードでのみ保持される, 400
バージョン 10.0.0 でサポート対象外, 193

カスタム照合作成ウィザード
バージョン 10.0.0 でサポート対象外, 193

カタログ
バージョン 10.0.0 での動作の変更, 176
バージョン 11.0.0 での動作の変更, 58

カラム圧縮
バージョン 10.0.0 の新機能, 133

カラム属性
再構築されたデータベースで次に使用可能な値
を保持, 397

カラム名の送信
バージョン 10.0.0 での動作の変更, 209

環境変数
コマンド・シェル, xii
コマンド・プロンプト, xii
バージョン 10.0.0 での動作の変更, 171
バージョン 10.0.0 の強化, 163

監査
バージョン 10.0.0 での動作の変更, 169
バージョン 10.0.0 の強化, 134
バージョン 11.0.0 の強化, 34
バージョン 11.0.0 の新機能, 89

管理ツール
11.0.1 での動作の変更, 12
バージョン 10.0.0 での動作の変更, 256
バージョン 11.0.1 の強化, 8

カーソル
バージョン 10.0.0 での Ultra Light の強化, 238

き

規格
508 条準拠, 364

規格と互換性
508 条準拠, 364

機能の統計の収集
バージョン 10.0.0 の新機能, 255

キャッシュ
バージョン 10.0.0 での動作の変更, 172
バージョン 9.0.1 の強化, 294

キャッシュ・ウォーミング
バージョン 9.0.1 の新機能, 289

キャッシュ・サイズ
バージョン 10.0.0 での Ultra Light の動作の変
更, 239
バージョン 10.0.0 での動作の変更, 172
バージョン 9.0.1 の強化, 294

キャッシュ：スカベンジ・アクセス統計値
バージョン 10.0.0 の新機能, 146

キャッシュ：スカベンジ統計値
バージョン 10.0.0 の新機能, 146

キャッシュ：パニック統計値
バージョン 10.0.0 の新機能, 146

キャッシュ・ページ：空き統計値
バージョン 10.0.0 の新機能, 146

キャッシュ・ページ：ファイル・ダーティ統計値
バージョン 10.0.0 の新機能, 146

キャッシュ・ページ：ファイル統計値
バージョン 10.0.0 の新機能, 146

キャッシュ・ページ：割り当て構造体統計値
バージョン 10.0.0 の新機能, 146

キャッシュ：マルチページ割り当て統計値
バージョン 10.0.0 の新機能, 146

キャッシュ読み込み：ワーク・テーブル統計値
バージョン 11.0.0 の新機能, 41

行番号
Interactive SQL バージョン 11.0.0 の強化, 93

強力な暗号化
バージョン 10.0.1 の強化, 103

近接検索
11.0.1 での動作の変更, 9

キー・ジョイン
バージョン 10.0.0 での動作の変更, 191

キー・ペア・ジェネレータ・ユーティリティ
[createkey]
バージョン 11.0.0 の新機能, 76

キーボード・ショートカット

バージョン 11.0.0 の新機能, 88

く

クイック・スタート

バージョン 11 へのデータベースのアップグレード, 396

クエリ・エディタ

バージョン 9.0.1 の強化, 286

クエリ最適化

バージョン 11.0.1 の強化, 6

クエリ内並列処理

バージョン 10.0.0 の新機能, 129

バージョン 10.0.1 での動作の変更, 109

クエリ・パフォーマンス

バージョン 11.0.1 の強化, 6

クライアントでの文のキャッシュ

バージョン 10.0.1 での動作の変更, 111

バージョン 10.0.1 の新機能, 103

クライアント・ネットワーク・レイヤ

バージョン 10.0.0 の Ultra Light の新機能, 240

クライアント・ファイル、書き込み

バージョン 11.0.0 の新機能, 30

クライアント・ファイル、読み込み

バージョン 11.0.0 の新機能, 30

クライアント・メッセージ・ストア

バージョン 10.0.0 での QAnywhere によるトランザクション・ログの使用の停止, 229

バージョン 11.0.0 の強化, 79

クライアント・メッセージ・ストア ID

バージョン 10.0.0 での動作の変更, 228

クラスタード・ハッシュ GROUP BY

バージョン 9.0.1 の新機能, 289

グラフィカルなプラン

バージョン 10.0.0 での動作の変更, 251

クリエータ ID

バージョン 10.0.0 の Ultra Light での強化, 237

グループ化されたコミット・フラッシュ

バージョン 10.0.1 での Ultra Light の強化, 122

グローバル・テンポラリ・テーブル

バージョン 10.0.0 の強化, 155

け

計算カラム

バージョン 11.0.0 での動作の変更, 63

結果

バージョン 9.0.1 の Interactive SQL の強化, 292

バージョン 9.0.1 の Sybase Central の強化, 292

読み込み専用に指定、バージョン 11.0.0 の新機能, 91

[結果] ウィンドウ枠

バージョン 11.0.0 の強化, 93

結果セット

読み込み専用に指定、バージョン 11.0.0 の新機能, 91

結果セットからの SQL 文の生成

バージョン 11.0.0 の強化, 93

結果セットの編集の無効化

バージョン 11.0.0 の新機能, 91

権限

継承、バージョン 11.0.0 の新機能, 35

言語選択ユーティリティ [dblang]

バージョン 10.0.1 での動作の変更, 111

バージョン 9.0.1 の廃止予定機能, 301

検証

バージョン 10.0.0 での動作の変更, 136

バージョン 11.0.0 での動作の変更, 36

検証ユーティリティ [dbvalid]

バージョン 10.0.0 の強化, 139

バージョン 11.0.0 の強化, 36

バージョン 9.0.1 での動作の変更, 302

バージョン 9.0.1 の強化, 288

こ

高可用性

バージョン 10.0.0 の新機能, 129, 130

交換アルゴリズム

バージョン 10.0.0 の新機能, 129

高速ランチャ

バージョン 11.0.0 での強化, 88

互換性

Ultra Light ソフトウェアのアップグレード, 419

Ultra Light のアップグレード, 419

クライアント/サーバ, 392

データベースとデータベース・サーバ, 392

問題, 392

コマンド・シェル

引用符, xii

カッコ, xii

環境変数, xii

中カッコ, xii

表記規則, xii

コマンド・プロンプト

引用符, xii

カッコ, xii
環境変数, xii
中カッコ, xii
表記規則, xii
コマンド・ライン・ユーティリティ
 Ultra Light でのアップグレード, 419
 Ultra Light の複数バージョン, 419
 アップグレード, 394
 複数バージョン, 394
コミット
 バージョン 10.0.1 での Ultra Light の強化, 122
コンソール・ユーティリティ [dbconsole]
 バージョン 10.0.0 の強化, 250
 バージョン 9.0.1 の強化, 291
コールバック関数
 バージョン 9.0.2 の新機能, 263

さ

再起動可能なダウンロード
 9.0.1 の Ultra Light の新機能, 298
再構築
 アップグレードを行う前の注意事項, 395
 制限, 399
 トラブルシューティング, 403
 バージョン 11 のデータベース, 396, 399, 405
再構築の失敗
 バージョン 11 へのデータベースのアップグレード, 403
サイレント・インストール
 バージョン 11.0.0 での動作の変更, 71
サブクエリ
 9.0.1 の Ultra Light の新機能, 297
サポート
 ニュースグループ, xiv
サポートされるプラットフォーム
 バージョン 11.0.0 での動作の変更, 66
サポート・ユーティリティ
 バージョン 11.0.0 の強化, 99
サポート・ユーティリティ [dbsupport]
 バージョン 10.0.0 の新機能, 139, 255
 バージョン 10.0.1 の強化, 109
参照整合性
 バージョン 10.0.0 の Ultra Light の新機能, 239
サンプル
 バージョン 10.0.0 での新しいデフォルトのインストール・ロケーション, 256
サンプル・データベース

SQL Anywhere バージョン 11.0.0 の強化, 99
 バージョン 10.0.0 での名前の変更, 257
サーバ・エラー・コード
 バージョン 9.0.0 での Mobile Link の動作の変更, 329
サーバ側の転送ルール
 バージョン 10.0.0 の強化, 227
サーバ側のバックアップ
 バージョン 11.0.1 の強化, 7
サーバ管理要求
 バージョン 10.0.0 の QAnywhere の新機能, 227
 バージョン 11.0.0 の強化, 79
サーバ起動同期
 バージョン 10.0.0 の追加機能, 212
サーバ起動同期の Listener
 バージョン 11.0.0 の強化, 76
サーバ・グループ
 バージョン 10.0.0 での Mobile Link の新機能, 207
サーバ・ファームでのサーバ起動同期
 バージョン 11.0.0 の強化, 76
サーバ・プロパティ
 バージョン 10.0.0 での動作の変更, 166
 バージョン 11.0.0 の新機能, 39
サーバ・プロパティ・ファイル
 バージョン 10.0.0 で廃止予定の QAnywhere の機能, 229
サーバ名
 バージョン 10.0.0 での動作の変更, 168
サーバ・メッセージ
 バージョン 11.0.0 の強化, 55
サーバ・ライセンス・ユーティリティ [dblic]
 バージョン 10.0.0 での動作の変更, 194
 バージョン 10.0.1 での動作の変更, 124
サーバ列挙ユーティリティ [dblocate]
 バージョン 10.0.0 での動作の変更, 171
 バージョン 10.0.0 の新機能, 138
 バージョン 9.0.1 の強化, 290
サービス
 バージョン 10.0.0 の強化, 138
 バージョン 9.0.1 の強化, 293
サービス作成ウィザード
 バージョン 11.0.1 の強化, 22
 バージョン 9.0.1 の強化, 293
サービスとしてログインする権限
 dbsvc ユーティリティで付与, 171
[サービスのプロパティ] ウィンドウ

バージョン 9.0.1 の強化, 293
サービス・ユーティリティ [dbsvc]
バージョン 10.0.0 での動作の変更, 171
バージョン 10.0.0 の強化, 138, 159
バージョン 11.0.1 の強化, 3

し

識別子
バージョン 10.0.0 での動作の変更, 168
バージョン 9.0.1 での動作の変更, 303
システム・テーブル
バージョン 10.0.0 での動作の変更, 176
バージョン 11.0.0 での動作の変更, 58
システム・パス
Ultra Light のユーティリティ, 419
ユーティリティ, 394
システム・ビュー
バージョン 10.0.0 での動作の変更, 176
バージョン 11.0.0 での動作の変更, 58
実行プラン
バージョン 11.0.0 の強化, 56
バージョン 11.0.1 の強化, 6
失敗したダウンロードの再開
9.0.1 の新機能 dbmlsync, 295
自動停止
バージョン 11.0.0 での動作の変更, 64
述部
バージョン 10.0.0 の Ultra Light の新機能, 239
昇格操作エージェンツ
バージョン 10.0.1 の新機能, 125
照合
Ultra Light アップグレードの推奨事項, 420
バージョン 10.0.0 での Ultra Light の強化, 235
バージョン 10.0.0 の新機能, 163
バージョン 8.0.0 で廃止予定, 386
バージョン 9.0.1 の新機能, 291
照合の適合化
バージョン 10.0.1 の新機能, 105
照合ユーティリティ (dbcollat)
バージョン 10.0.0 でサポート対象外, 193
詳細情報の検索／テクニカル・サポートの依頼
テクニカル・サポート, xiv
情報ユーティリティ [dbinfo]
バージョン 10.0.0 の強化, 137
証明書
取得場所, 345
バージョン 11.0.0 での動作の変更, 100

初期化ユーティリティ [dbinit]
バージョン 10.0.0 での -b オプションの動作の変更, 166
バージョン 10.0.0 での動作の変更, 170
バージョン 10.0.0 の強化, 137
バージョン 10.0.1 の強化, 109
バージョン 9.0.1 の強化, 288
ショートカット
バージョン 11.0.0 の新機能, 91
シリアル・プロトコル
8.0.2 から Mobile Link でのサポート終了, 345
新機能
バージョン 10.0.0, 127
バージョン 10.0.1, 101
バージョン 11.0.0, 27
バージョン 11.0.1, 1
バージョン 8.0.0, 360
バージョン 8.0.1, 350
バージョン 8.0.2, 334
バージョン 9.0.0, 306
バージョン 9.0.1, 286
バージョン 9.0.2, 260
シングル・ステップ
バージョン 11.0.0 の新機能, 91
進行オフセット
バージョン 10.0.0 での動作の変更, 221
診断ディレクトリ
バージョン 10.0.0 の新機能, 255
診断トレーシング
バージョン 10.0.0 の新機能, 130
シーケンス番号
バージョン 10.0.0 での Mobile Link システム・
テーブル・スキーマの変更, 203

す

数値データ型
バージョン 10.0.0 での動作の変更, 174
スキーマ
Ultra Light アップグレード, 424
Ultra Light アップグレードによる影響, 420
バージョン 10.0.0 の Ultra Light の新機能, 233
スキーマのアップグレード
9.0.1 の Ultra Light の強化, 298
スキーマのアップグレードの監視
9.0.1 の Ultra Light の強化, 298
スキーマ・ペインタ

- バージョン 10.0.0 で削除された Ultra Light, 245
- スクリプト
 - アップグレード, 417
- スクリプト化されたアップロード
 - バージョン 10.0.0 の新機能, 209
- スクリプト実行 [dbrunsql] ユーティリティ
 - バージョン 10.0.0 の新機能, 158
- スクリプトの無視
 - バージョン 11.0.1 の MobiLink (mlsrv11) の新機能, 14
- スクリプト・バージョン
 - バージョン 9.0.1 で設定可能, 294
- スケジュール作成ウィザード
 - バージョン 10.0.0 の新機能, 249
- ストアド・プロシージャ
 - バージョン 10.0.0 の強化, 155
- スナップショット・アイソレーション
 - バージョン 10.0.0 の Mobile Link サポート, 206
 - バージョン 10.0.0 の新機能, 130
- スマートフォン
 - Ultra Light バージョン 11.0.0 での強化, 86
 - バージョン 11.0.0 の強化, 99

せ

- 正規表現
 - バージョン 11.0.0 の新機能, 29
 - バージョン 11.0.1 での動作の変更, 10
- 制限
 - バージョン 11.0.0 での増加, 56
- 正常性と統計情報
 - バージョン 11.0.0 の新機能, 89
- 静的型 Java API サポート
 - バージョン 10.0.0 で削除された Ultra Light, 245
- 制約
 - バージョン 10.0.0 の強化, 162
- セキュリティ
 - Ultra Light バージョン 11.0.0 での強化, 84
 - バージョン 10.0.0 での Mobile Link の強化, 211
 - バージョン 10.0.0 での Ultra Light の強化, 235
- セキュリティ保護された機能
 - バージョン 10.0.0 の新機能, 136
- 接続
 - 接続オブジェクトを制御するパラメータの選択, 420
 - バージョン 11.0.0 での動作の変更, 64

- 接続 ID
 - バージョン 9.0.1 の強化, 293
- 接続アシスタント
 - バージョン 11.0.0 の新機能, 88
- [接続] ウィンドウ
 - バージョン 11.0.0 での強化, 88
 - バージョン 11.0.0 の新機能, 88
 - バージョン 9.0.1 での動作の変更, 302
- 接続コード
 - Ultra Light アップグレード, 424
- 接続数
 - バージョン 10.0.1 での Ultra Light の強化, 121
- 接続数の増加
 - バージョン 10.0.1 での Ultra Light の強化, 121
- 接続パラメータ
 - Ultra Light API の制御, 420
 - バージョン 10.0.0 での動作の変更, 169
 - バージョン 10.0.0 の強化, 134
- 接続プロパティ
 - バージョン 10.0.0 での動作の変更, 166
 - バージョン 11.0.0 の新機能, 38
- 接続プロファイル
 - バージョン 10.0.0 の強化, 248
- 接続文字列
 - バージョン 10.0.0 での動作の変更, 169
 - バージョン 10.0.0 の強化, 134
- 接続文字列をクリップボードにコピー
 - バージョン 11.0.0 の新機能, 88
- 設定可能なコミット・フラッシュ
 - バージョン 10.0.1 での Ultra Light の強化, 122
- 設定ファイル
 - バージョン 11.0.0 の強化, 36
- 全文検索
 - 11.0.1 での動作の変更, 9
 - バージョン 11.0.0 の新機能, 29

そ

- 送信先エイリアス
 - バージョン 10.0.0 の QAnywhere の新機能, 227
- 即時ビュー
 - バージョン 11.0.0 の新機能, 31
- ソート・ブロック・アルゴリズム
 - バージョン 10.0.0 で削除, 199

た

- 対象ユーザに対するログの冗長性
 - バージョン 11.0.1 の強化, 15

代替サーバ名

バージョン 10.0.0 の新機能, 129

ダイレクト・ページ・スキャン

Ultra Light バージョン 11.0.0 の新機能、テーブル API, 85

ダイレクト・ロー・ハンドリング

バージョン 10.0.0 での Mobile Link の新機能, 202

ダウンロード確認

バージョン 11.0.0 での動作の変更, 76

ダウンロード専用のパブリケーション

バージョン 10.0.0 の新機能, 210

多言語リソース開発キット

バージョン 11.0.0 での動作の変更, 67

タスク・リスト

バージョン 10.0.0 の新機能, 248

ち

チェックサム

バージョン 10.0.0 の強化, 135

バージョン 11.0.0 での動作の変更, 64

バージョン 11.0.0 の強化, 53

バージョン 9.0.1 の新機能, 288

チェックポイント

オートインクリメント・カラムのあるテーブル, 397

オートインクリメント・カラムのあるテーブルのデータ, 397

バージョン 10.0.0 の強化, 132

チェックポイント操作

バージョン 10.0.1 での Ultra Light の強化, 122

致命的なエラー

バージョン 10.0.0 の UNIX の強化, 159

注意事項

アップグレード, 395

抽出ユーティリティ

バージョン 11.0.0 での動作の変更, 67

バージョン 11.0.0 の強化, 80

重複するテキスト・インデックス

11.0.1 での動作の変更, 9

つ

通信：受信要求数統計値

バージョン 10.0.0 の新機能, 146

ツールバー

Interactive SQL バージョン 11.0.0 の強化, 93

て

ディスクが満杯

バージョン 9.0.2 の新機能, 263

ディスク読み込み：ワーク・テーブル統計値

バージョン 11.0.0 の新機能, 41

ディスク領域

バージョン 9.0.2 の新機能, 263

ディレクトリ・アクセス・サーバ

バージョン 10.0.0 の新機能, 163

テキスト・インデックス

11.0.1 での動作の変更, 9

バージョン 11.0.0 の新機能, 29

テキスト設定オブジェクト

バージョン 11.0.0 の新機能, 29

テキスト・プラン

Interactive SQL バージョン 11.0.0 で廃止予定の機能, 93

テキスト補完

バージョン 10.0.0 の新機能, 249

適用式

バージョン 11.0.0 の新機能, 47

テクニカル・サポート

ニュースグループ, xiv

[デッドロック] タブ

バージョン 10.0.0 の新機能, 248

デバイス

Ultra Light バージョン 11.0.0 での強化, 84

バージョン 10.0.0 での Ultra Light の強化, 234, 238

バージョン 10.0.1 での Ultra Light の強化, 121

デバッグ・モード

バージョン 11.0.0 の強化, 92

デフォルト・データベース・サーバ

バージョン 11.0.1 の強化, 7

デベロッパー・コミュニティ

ニュースグループ, xiv

展開

バージョン 10.0.0 の新しいウィザード, 202

ライト・ファイル、バージョン 10.0.0 でサポート対象外, 194

テンポラリ・ストアド・プロシージャ

バージョン 10.0.0 の新機能, 155

テンポラリ・テーブル

バージョン 10.0.0 の強化, 155

バージョン 10.0.1 での動作の変更, 113

バージョン 11.0.0 での動作の変更, 67, 68

テンポラリ・ファイル

- バージョン 10.0.0 の強化, 141
- バージョン 11.0.0 の UNIX に関する強化, 53
- データ型変換
 - バージョン 10.0.1 での動作の変更, 112
- データ・ソース・ユーティリティ [dbdsn]
 - バージョン 10.0.0 での動作の変更, 193
 - バージョン 10.0.0 の新機能, 137
 - バージョン 10.0.1 の強化, 109
- データのロード
 - バージョン 10.0.1 での動作の変更, 113
- データベース
 - SQL Anywhere 10.0.0 でサポートされるバージョン, 164
 - 圧縮、バージョン 10.0.0 でサポート対象外, 194
 - アップグレード, 392
 - バージョン 10.0.0 以降からのファイル・フォーマットのアップグレード, 405
 - バージョン 9 以前からのファイル・フォーマットのアップグレード, 399
 - ライトファイル、バージョン 10.0.0 でサポート対象外, 194
- データベース圧縮ウィザード
 - バージョン 10.0.0 でサポート対象外, 195
 - バージョン 9.0.1 の強化, 293
- データベース・アップグレード・ウィザード
 - 使用, 405
 - バージョン 10.0.0 での動作の変更, 165
 - バージョン 9.0.1 の強化, 293
- データベース暗号化
 - バージョン 11.0.0 の強化, 34
- データベース・アンロード・ウィザード
 - バージョン 10.0.0 での動作の変更, 173
 - バージョン 9 以前のデータベースの再構築, 401
- データベース・オプション
 - バージョン 10.0.0 での動作の変更, 174
 - バージョン 11.0.0 の強化, 30
- データベース検証ウィザード
 - バージョン 11.0.0 の新機能, 90
 - バージョン 9.0.1 の強化, 293
- データベース・サイズ
 - バージョン 11.0.0 の強化, 55
- データベース作成ウィザード
 - バージョン 10.0.1 の強化, 107
 - バージョン 11.0.0 の強化, 92
- データベース・サーバ
 - バージョン 10.0.0 の強化, 138
- データベース・サーバ・メッセージ・ウィンドウ
 - バージョン 10.0.0 の強化, 163
- データベース抽出ウィザード
 - Ultra Light での使用, 426
 - Ultra Light バージョン 11.0.0 での強化, 83
 - バージョン 10.0.0 での動作の変更, 173
- データベース・ツールのインポート・ライブラリ
 - 11.0.0 の新機能, 62
- データベース展開ウィザード
 - バージョン 10.0.0 でサポート対象外, 195
- データベース・ドキュメント・ウィザード
 - バージョン 11.0.0 の新機能, 89
- データベース・ドキュメントの生成
 - バージョン 11.0.0 の新機能, 89
- データベース内の Java
 - バージョン 10.0.0 での動作の変更, 169
- データベースのアップグレード
 - Sybase Central, 405
 - Ultra Light アップグレード・パス, 421
 - Ultra Light での考慮事項, 420
 - 制限, 399
 - バージョン 9.0.2 で廃止予定の古いフォーマット, 277
- データベースのアップグレード・ウィザード
 - Ultra Light バージョン 11.0.0 の例, 422
- データベースのアンロード
 - バージョン 10.0.0 の強化, 158
- データベースの再構築
 - 再構築の失敗, 403
 - バージョン 10.0.0 で必要, 165
 - バージョン 10.0.0 の強化, 158
 - バージョン 11.0.0 の強化, 32
- データベースのドキュメント化
 - バージョン 11.0.0 の新機能, 89
- データベースのミラーリング
 - バージョン 10.0.0 の新機能, 129
- データベース・バックアップ・ウィザード
 - バージョン 9.0.1 の強化, 293
- データベース・ファイル・フォーマット
 - Ultra Light のアップグレード, 420
 - アップグレード, 399, 405
 - バージョン 9.0.2 で廃止予定の古いフォーマット, 277
- データベース・プロパティ
 - バージョン 10.0.0 での動作の変更, 166
 - バージョン 11.0.0 の新機能, 40

データベース・プロパティ (Ultra Light)

9.0.1 の新機能, 299

バージョン 10.0.0 での強化, 235

データベース・ミラーリング

EBF の適用, 407

アップグレード, 407

バージョン 10.0.1 の強化, 107

バージョン 11.0.0 での動作の変更, 64

バージョン 11.0.0 の強化, 32

データベース・リストア・ウィザード

バージョン 9.0.1 の強化, 293

テーブル API

Ultra Light バージョン 11.0.0 での強化, 85

テーブル暗号化

バージョン 11.0.0 の強化, 34

テーブルの暗号化

バージョン 10.0.0 の新機能, 133

テーブルの順序

バージョン 10.0.0 の Ultra Light の新機能, 240

テーブルの編集の無効化

バージョン 11.0.0 の新機能, 91

テーブル・マッピング追加ウィザード

バージョン 10.0.1 で削除, 116

と

問い合わせ

BLOB、バージョン 11.0.0 の新機能, 32

ファイル、バージョン 11.0.0 の新機能, 32

同期

バージョン 10.0.0 での Ultra Light の動作の変更, 239

同期 ID

バージョン 10.0.0 での Mobile Link の新機能, 206

同期オブザーバ

9.0.1 の Ultra Light の強化, 298

同期ストリーム

バージョン 10.0.0 の Ultra Light の改訂された機能, 235

同期ストリームのセキュリティ

バージョン 11.0.0 の新機能, 75

同期プロファイル

Ultra Light バージョン 11.0.0 での強化, 81

バージョン 11.0.0 の新機能, 75

同期プロファイル作成ウィザード

バージョン 11.0.0 の新機能, 90

同期モデル作成ウィザード

バージョン 10.0.0 の Mobile Link の新機能, 201

同期モデル展開ウィザード

バージョン 10.0.0 の Mobile Link の新機能, 201

バージョン 11.0.0 の強化, 92

統計

バージョン 10.0.0 での Mobile Link の動作の変更, 218

統合化ログイン

バージョン 9.0.1 の強化, 291

統合データベース

アップグレード, 411

動作の変更

バージョン 10.0.0, 127

バージョン 10.0.1, 101

バージョン 11.0.0, 27

バージョン 11.0.1, 1

バージョン 8.0.0, 360, 382

バージョン 8.0.1, 350, 356

バージョン 8.0.2, 334, 344

バージョン 9.0.0, 306, 325

バージョン 9.0.1, 286, 301

バージョン 9.0.2, 260, 277

同時接続数

バージョン 10.0.1 での Ultra Light の強化, 121

動的 SQL

バージョン 10.0.0 の Ultra Light の新機能, 234

動的キャッシュ・サイズ決定

バージョン 10.0.0 の強化, 158

動的トラップ

バージョン 10.0.1 の強化, 109

独立性レベル

Ultra Light ReadUncommitted、バージョン 11.0.0 での強化, 82

トピック

グラフィック・アイコン, xiii

ドメイン

バージョン 10.0.0 での動作の変更, 171

トラブルシューティング

Ultra Light のコマンド・ライン・ユーティリティ, 419

コマンド・ライン・ユーティリティ, 394

ニュースグループ, xiv

バージョン 11 へのデータベースのアップグレード, 403

トランザクション

9.0.1 の Ultra Light 動的 SQL の新機能, 297

トランザクション・ログ

バージョン 10.0.0 の強化, 135
バージョン 11.0.0 の強化, 30
バージョン 9.0.2 の新機能, 264
トランザクション・ログのオフセット
データベース・ファイルのアップグレード,
399, 405
トランスポート・レイヤ・セキュリティ
証明書の取得場所, 345
トリガ
バージョン 10.0.1 の強化, 108
トリガ作成ウィザード
バージョン 10.0.1 の強化, 107

な

内部アンロード
カスタム照合の保持, 400
名前付きスクリプト・パラメータ
バージョン 10.0.0 の Mobile Link の新機能, 205
名前付きパイプ
バージョン 10.0.0 でサポート対象外, 193

に

ニュースグループ
テクニカル・サポート, xiv

ね

ネスト・ブロック・ジョイン・アルゴリズム
バージョン 10.0.0 で削除, 199
[ネットワーク] タブ
バージョン 11.0.0 の新機能, 88
ネットワーク・プロトコル
バージョン 10.0.0 の Mobile Link の改訂された
機能, 212
バージョン 10.0.0 の Ultra Light の改訂された機
能, 235
ネットワーク・レイヤ
バージョン 10.0.0 の Mobile Link の向上した機
能, 206
バージョン 10.0.0 の Ultra Light Mobile Link ク
ライアントの向上した機能, 240

の

ノルウェー語
バージョン 10.0.0 の新機能, 163

は

廃止予定機能

バージョン 10.0.0, 127
バージョン 8.0.0, 360
バージョン 8.0.1, 350, 356
バージョン 8.0.2, 334
バージョン 9.0.0, 306, 328
バージョン 9.0.1, 286, 301
バージョン 9.0.2, 260, 277
廃止予定の機能
バージョン 10.0.1, 101
バージョン 11.0.0, 27
バージョン 11.0.1, 1
ハイフン
11.0.1 での全文検索の動作の変更, 9
バグ
フィードバックの提供, xiv
パケット・サイズ
バージョン 11.0.0 でのデフォルトの拡大, 55
パス
Ultra Light アプリケーションのアップグレー
ド, 425
Ultra Light データベースのアップグレード,
421
パススルー・スクリプト作成ウィザード
バージョン 11.0.0 の新機能, 90
パススルー・ダウンロード作成ウィザード
バージョン 11.0.0 の新機能, 90
パスワード
バージョン 10.0.0 での Ultra Light の動作の変
更, 235
バージョン 10.0.0 での動作の変更, 165
バージョン 11.0.0 の強化, 33
バージョン 9.0.1 での動作の変更, 302
パスワードのハッシュ処理
バージョン 10.0.0 での Ultra Light の動作の変
更, 235
バージョン 10.0.0 での動作の変更, 165
派生テーブル
9.0.1 の Ultra Light の新機能, 297
バージョン 10.0.0 でのキー・ジョインの動作の
変更, 191
バックアップ
バージョン 11.0.1 の強化, 7
バックアップ・イメージ作成ウィザード
バージョン 9.0.1 の強化, 293
バックアップとリカバリ
バージョン 11.0.0 の強化, 33
バックアップ・ユーティリティ [dbbackup]

-
- バージョン 10.0.0 の強化, 135
 - バージョン 9.0.1 の強化, 290
 - バックグラウンド同期
 - Ultra Light バージョン 11.0.0 での強化, 83
 - バックログ・オプション
 - バージョン 10.0.0 で削除, 217
 - ハッシュ用 SHA256 アルゴリズム
 - バージョン 10.0.0 の新機能, 151
 - バッチ
 - バージョン 11.0.0 の強化, 54
 - パフォーマンス
 - Ultra Light アプリケーションのアップグレード, 424
 - Ultra Light データベースのアップグレード, 420
 - データベース・ファイルのアップグレード, 399, 405
 - バージョン 10.0.0 での Ultra Light の強化, 233
 - バージョン 10.0.0 の強化, 132
 - パフォーマンス・モニタ
 - バージョン 10.0.0 の強化, 146
 - バージョン 11.0.0 の強化, 38
 - バージョン
 - 同期スクリプトのアップグレード, 417
 - バージョン 10.0.0
 - 新機能, 127
 - 動作の変更, 127
 - バージョン 10.0.0 の新機能
 - 概要, 127
 - バージョン 10.0.1
 - 新機能, 101
 - 動作の変更, 101
 - バージョン 11.0.0
 - アップグレード, 391
 - アップグレードの既知の問題, 404
 - 新機能, 27
 - 動作の変更, 27
 - バージョン 11.0.0 の新機能
 - 概要, 27
 - バージョン 11.0.0 へのアップグレード
 - 説明, 391
 - バージョン 11.0.1
 - 新機能, 1
 - 動作の変更, 1
 - バージョン 11.0.1 の新機能
 - 概要, 1
 - バージョン 5
 - SQL Remote インストール環境のアップグレード, 429
 - バージョン 8.0.0
 - Adaptive Server Anywhere の新機能, 360
 - Mobile Link の新機能, 374
 - 新機能, 360
 - 動作の変更, 360, 382, 388
 - バージョン 8.0.1
 - 新機能, 350
 - 動作の変更, 350, 356
 - バージョン 8.0.2
 - 新機能, 334
 - 動作の変更, 334, 344
 - バージョン 9
 - 動作の変更, 325
 - バージョン 9.0.0
 - 新機能, 306
 - 動作の変更, 306
 - バージョン 9.0.1
 - 新機能, 286
 - 動作の変更, 286, 301
 - バージョン 9.0.2
 - 新機能, 260
 - 動作の変更, 260, 277
 - バージョン 9 以前のデータベースの再構築
 - 説明, 399
 - バージョンごとの変更箇所
 - 10.0.0, 127
 - 10.0.1, 101
 - 11.0.0, 27
 - 11.0.1, 1
 - 8.0.0, 360
 - 8.0.1, 350, 356
 - 8.0.2, 334, 344
 - 9.0.0, 306, 325
 - 9.0.1, 286, 301
 - 9.0.2, 260, 277
 - バージョン・ストア・ページ統計値
 - バージョン 10.0.0 の新機能, 146
 - パーソナル・サーバ
 - バージョン 10.0 で変更されたデフォルト TCP/IP アドレス, 172
- ## ひ
- ヒストグラム・ユーティリティ [dbhist]
 - バージョン 10.0.0 の強化, 137
 - ビッグ・エンディアン UTF-16 エンコード

- バージョン 11.0.0 の新機能, 54
- 日付
 - バージョン 10.0.0 の強化, 163
- ビット配列
 - バージョン 10.0.0 の新機能, 155
- 非同期 I/O
 - バージョン 9.0.1 の新機能, 293
- 非ブロッキング・ダウンロード確認、QAnywhere
 - バージョン 11.0.0 での動作の変更, 79
- 非ブロッキング・ダウンロード確認スクリプト
 - バージョン 11.0.0 の強化, 75
- ビュー
 - バージョン 11.0.0 の強化, 54
- ビューの依存性
 - バージョン 10.0.0 の新機能, 132
- ビューの一致
 - バージョン 10.0.0 の新機能, 130
- [ビューのプロパティ] ウィンドウ
 - バージョン 10.0.1 の強化, 107
- 表記規則
 - コマンド・シェル, xii
 - コマンド・プロンプト, xii
 - マニュアル, x
 - マニュアルでのファイル名, xi
- 標準アップグレードの注意事項
 - 説明, 395
- ヒープ：カーバ統計値
 - バージョン 10.0.0 の新機能, 146
- ヒープ：クエリ処理統計値
 - バージョン 10.0.0 の新機能, 146
- ヒープ：再配置可能統計値
 - バージョン 10.0.0 の新機能, 146
- ヒープ：再配置可能ロック統計値
 - バージョン 10.0.0 の新機能, 146

ふ

- ファイル
 - 問い合わせ、バージョン 11.0.0 の新機能, 32
- ファイル・フォーマット
 - Ultra Light のアップグレード, 420
 - アップグレード, 399, 405
 - バージョン 10.0.0 の Ultra Light の新機能, 233
- ファイル名
 - .cdb 拡張子、バージョン 10.0.0 でサポート対象外, 194
 - .wrt 拡張子、バージョン 10.0.0 でサポート対象外, 194

- フィードバック
 - エラーの報告, xiv
 - 更新のご要望, xiv
 - 提供, xiv
 - マニュアル, xiv
- フェールオーバー、Mobile Link サーバ・ファーム
 - バージョン 11.0.0 の新機能, 73
- フォントの選択
 - バージョン 10.0.0 の新機能, 249
- 負荷分散、Mobile Link サーバ・ファーム
 - バージョン 11.0.0 の新機能, 73
- 複数の結果セット
 - バージョン 10.0.0 での動作の変更, 251
- 複数バージョン
 - Adaptive Server Anywhere, 394
 - Ultra Light, 419
- プライマリ・キー制約
 - バージョン 10.0.0 での動作の変更, 168
- プラグイン・モジュール
 - バージョン 10.0.0 の Ultra Light の新機能, 236
- プラットフォーム
 - バージョン 10.0.0 で削除された Ultra Light, 244
 - バージョン 10.0.0 での Ultra Light の強化, 234
 - バージョン 10.0.1 での Ultra Light の強化, 121
 - Ultra Light バージョン 11.0.0 での強化, 84
- ブランク埋め込み
 - バージョン 10.0.0 での動作の変更, 166
- ブランのキャッシュ
 - バージョン 10.0.1 の強化, 108
 - バージョン 11.0.0 の強化, 55
- ブラン・ビューワ
 - バージョン 11.0.0 での強化, 93
 - バージョン 11.0.0 で廃止予定の機能, 93
- プリフェッチ
 - バージョン 11.0.0 の強化, 33
- プロキシ・テーブル作成ウィザード
 - バージョン 9.0.1 の強化, 292
- プロキシ・ポート
 - バージョン 10.0.0 の強化, 158
- プロシージャ
 - バージョン 10.0.0 の強化, 147
- ブロック
 - バージョン 11.0.0 での動作の変更, 67
- プロトコルのオプション
 - バージョン 10.0.0 の強化, 134
- プロパティ

バージョン 10.0.0 での動作の変更, 166
プロパティ・ウィンドウ
Ultra Light バージョン 11.0.0 で廃止予定の機能, 92
プロパティ関数
バージョン 10.0.0 の強化, 149
プロファイリング・モード
バージョン 10.0.0 の新機能, 130
文キャッシュ・ヒットの統計値
バージョン 10.0.1 の新機能, 104
文キャッシュ・ミスの統計値
バージョン 10.0.1 の新機能, 104
分散読み込み
バージョン 10.0.0 での動作の変更, 168

へ

並列インデックス・スキャン
バージョン 10.0.0 の新機能, 129
バージョン 9.0.1 の新機能, 289
並列テーブル・スキャン
バージョン 10.0.0 の新機能, 129
並列バックアップ
バージョン 10.0.0 の新機能, 135
バージョン 11.0.1 の強化, 7
ヘルプ
テクニカル・サポート, xiv
ヘルプへのアクセス
テクニカル・サポート, xiv
変換
バージョン 10.0.0 でのデータ型の動作の変更, 174
ページ・サイズ
バージョン 10.0.0 で変更されたデフォルト, 172

ほ

保護された機能
バージョン 11.0.0 での動作の変更, 63
ホスト・プロトコル・オプション
バージョン 10.0.0 での動作の変更, 172

ま

マテリアライズド・ビュー
アップグレードの考慮事項, 392
バージョン 10.0.0 の新機能, 130
バージョン 10.0.1 の強化, 108
バージョン 11.0.0 の強化, 31

マニュアル
SQL Anywhere, viii
表記規則, x
マニュアルの強化
バージョン 10.0.0, 254
バージョン 11.0.0, 98
バージョン 11.0.1, 26
バージョン 9.0.2, 276
マルチデータベースのサポート
9.0.1 の Ultra Light の新機能, 299
マルチテーブル・ジョイン
Ultra Light 動的 SQL, 323
マージ・モジュール
バージョン 10.0.0 での動作の変更, 158

み

ミラー・データベース
バージョン 11.0.0 の強化, 32
ミラーリング
EBF の適用, 407
アップグレード, 407
バージョン 11.0.0 の強化, 31

め

メッセージ
9.0.1 の Mobile Link の新機能, 294
メッセージ・ストア、Ultra Light
バージョン 11.0.0 の新機能, 78
メッセージ・ストア、アーカイブ
バージョン 11.0.0 の強化, 79
メッセージ・ストア、サーバ側の削除ルール
バージョン 11.0.0 の強化, 79
メッセージ・セレクト
バージョン 10.0.0 の QAnywhere の新機能, 226
メモリ・ページ：カーバ統計値
バージョン 10.0.0 の新機能, 146
メモリ・ページ：クエリ処理統計値
バージョン 10.0.0 の新機能, 146
メモリ・ページ：固定カーソル統計値
バージョン 10.0.0 の新機能, 146
メンテナンス・プラン
バージョン 10.0.0 の新機能, 249
メンテナンス・プラン作成ウィザード
バージョン 11.0.1 の強化, 22
メンテナンス・リリース

データベース・ミラーリング・システムへの適用, 407

も

文字セット

バージョン 10.0.0 での Ultra Light の強化, 235

文字セットの変換

バージョン 10.0.0 での動作の変更, 167

文字セット変換

バージョン 10.0.0 の強化, 131

文字長セマンティック

バージョン 10.0.0 の新機能, 155

モニタ

バージョン 11.0.1 の新機能, 25

モバイル Web サービス

バージョン 10.0.0 の新機能, 225

ゆ

ユニーク識別子

バージョン 10.0.0 の強化, 156

ユーザ ID

バージョン 10.0.0 での動作の変更, 168

ユーザ作成ウィザード

バージョン 11.0.0 の強化, 92

ユーザ定義の関数

バージョン 10.0.0 の強化, 147

ユーティリティ

Ultra Light の生成 [ulgen] ユーティリティ, 425

バージョン 10.0.1 での Ultra Light の動作の変更, 122

[ユーティリティ] タブ

バージョン 10.0.0 での動作の変更, 252

ユーティリティ・データベース

バージョン 10.0.0 での動作の変更, 199

バージョン 10.0.0 の新機能, 141

バージョン 11.0.0 での動作の変更, 63

よ

要求のロギング

バージョン 10.0.0 の強化, 164

バージョン 9.0.1 の強化, 291

要求レベル・ログ (参照 要求ログ)

読み込み専用

ミラー・データベースのアクセス、バージョン 11.0.0 の新機能, 32

ライト・ファイル、バージョン 10.0.0 でサポート対象外, 194

読み込み専用データベース

バージョン 10.0.0 での動作の変更, 169

バージョン 11.0.1 での動作の変更, 24

ライト・ファイル、バージョン 10.0.0 でサポート対象外, 194

予約語

バージョン 11 へのアップグレードのトラブルシューティング, 404

ら

ライセンス

バージョン 10.0.1 での動作の変更, 124

ライトウェイト・ポーリングのアクション変数

バージョン 11.0.1 の強化, 14

ライトウェイト・ポーリングのリスナ・キーワード

バージョン 11.0.1 の強化, 14

ライト・ファイル

バージョン 10.0.0 でサポート対象外, 194

ライト・ファイル作成ウィザード

バージョン 10.0.0 でサポート対象外, 195

り

リカバリ

バージョン 10.0.0 の強化, 135

リダイレクタ

9.0.1 での Apache Web サーバのサポート, 296

バージョン 10.0.0 の強化, 207

バージョン 11.0.0 で廃止予定, 75

リトル・エンディアン UTF-16 エンコード

バージョン 11.0.0 の新機能, 54

リファレンス・データベース

Ultra Light アップグレード, 421

リフレッシュ

マテリアライズド・ビュー、バージョン 11.0.0 での動作の変更, 64

リモート ID

バージョン 10.0.0 での Mobile Link の新機能, 208

リモート ID ファイル

バージョン 10.0.0 の新機能, 213

リモート・サーバ作成ウィザード

バージョン 9.0.1 の強化, 292

リモート・データ・アクセス

バージョン 10.0.0 での ODBC ドライバの変更, 258

バージョン 11.0.0 での動作の変更, 68

リモート・データベース
アップグレード, 415
リレー・サーバのホスト
バージョン 11.0.0 の新機能, 73

れ

連邦リハビリテーション法
508 条, 364

ろ

ロギング
バージョン 10.0.0 の強化, 154
ログイン・ポリシー
バージョン 11.0.0 の新機能, 29
ログ・ファイル
バージョン 10.0.0 の強化, 141
ログ変換ユーティリティ [dbtran]
バージョン 11.0.0 の強化, 37
ロック
Interactive SQL バージョン 11.0.0 の新機能, 90
バージョン 10.0.0 の強化, 132
バージョン 11.0.0 での動作の変更, 67
バージョン 11.0.1 での Interactive SQL の動作の
変更, 24
論理インデックス
バージョン 10.0.0 の新機能, 162
ロード
値から、バージョン 11.0.0 の新機能, 31
カラムから、バージョン 11.0.0 の新機能, 31
トランザクション・ログから、バージョン
11.0.0 の新機能, 31
ロー・トラバーサル
Ultra Light バージョン 11.0.0 の新機能, 85
ロールバック・ログ
バージョン 11.0.0 での動作の変更, 66

わ

ワイド文字
バージョン 10.0.0 での Ultra Light の強化, 241
ワーカ・スレッド
バージョン 10.0.0 での Mobile Link の動作の変
更, 216
